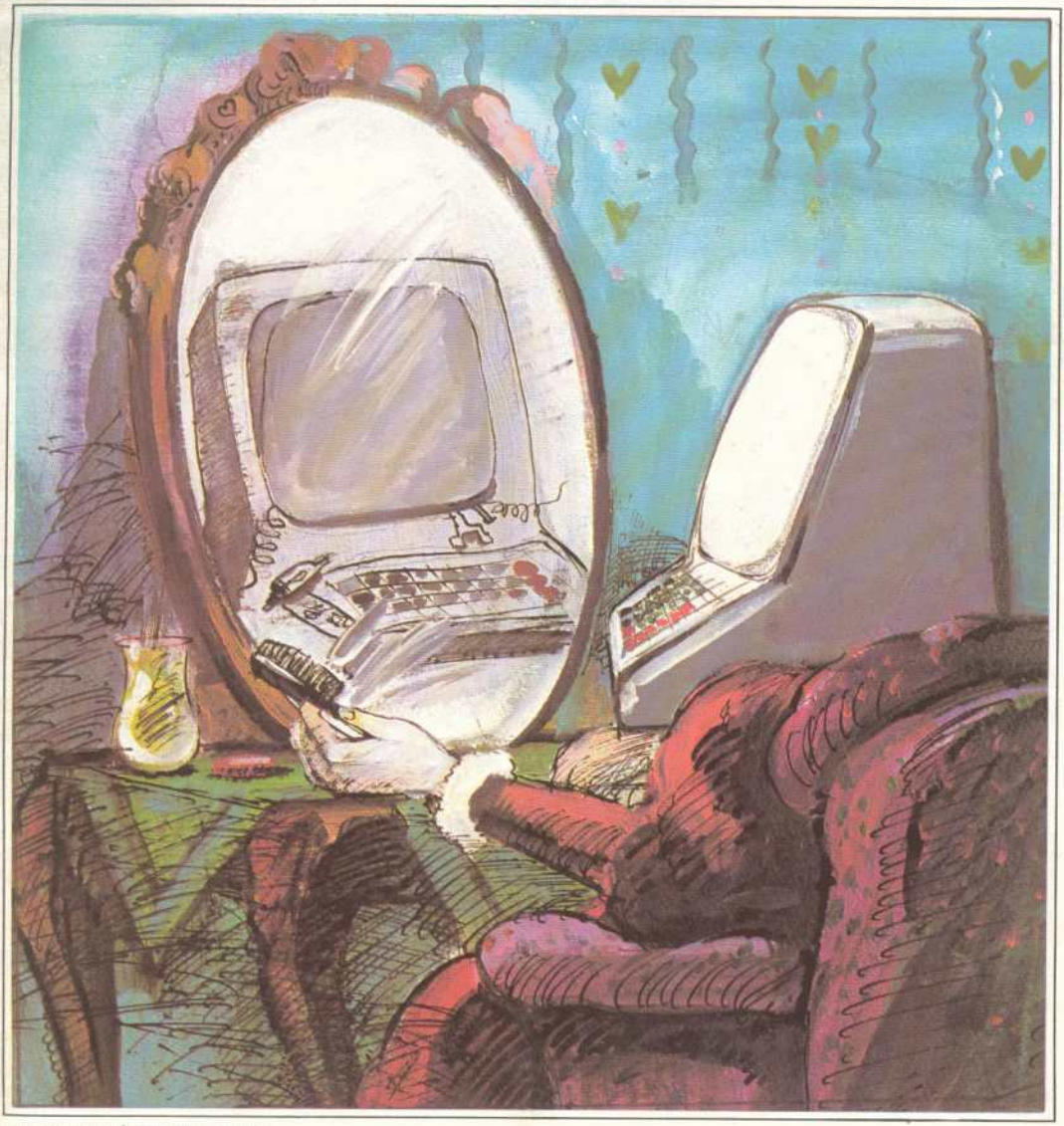
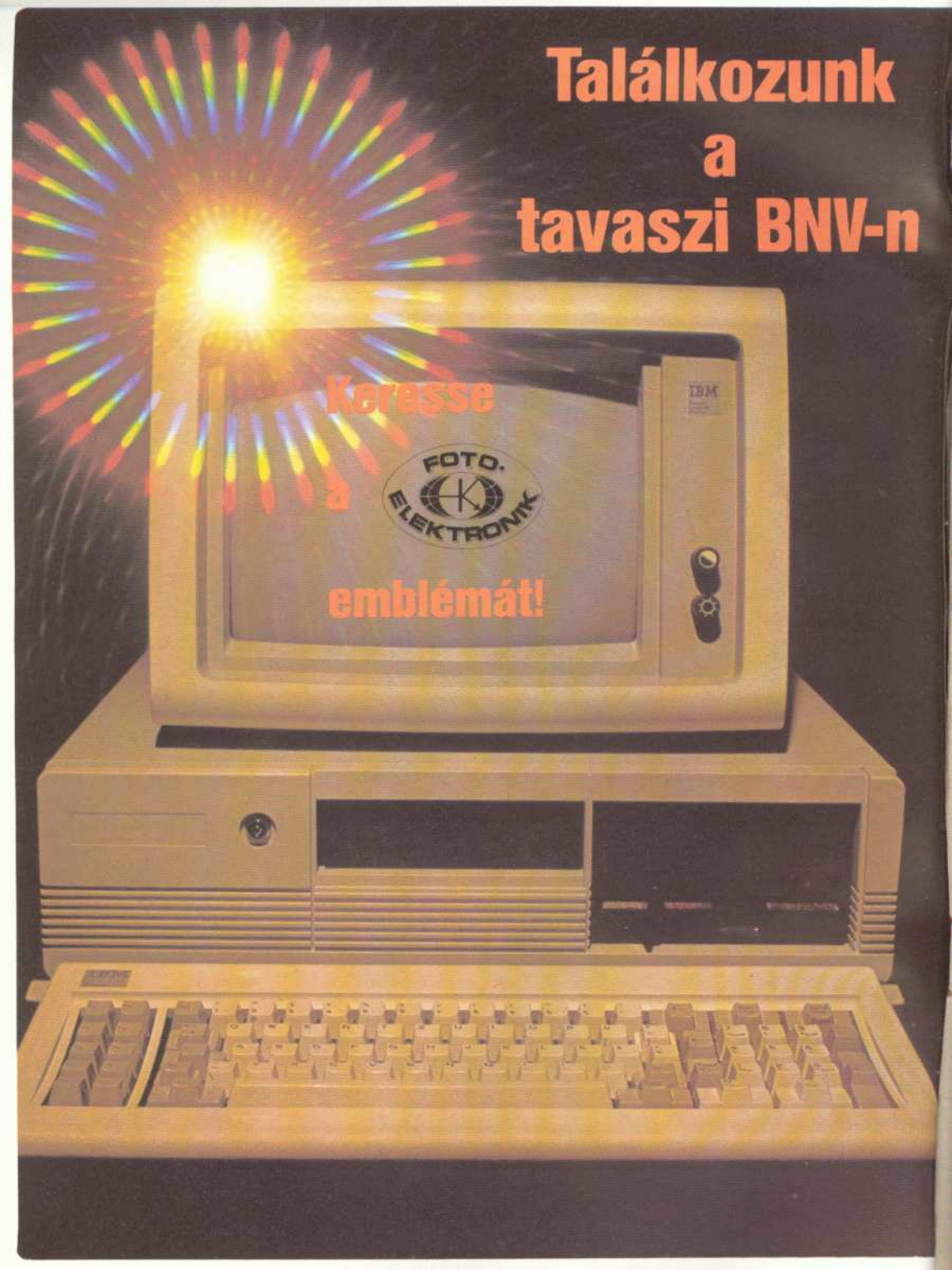
Ára: 30 Ft

μ mikro számítógép magazin



IBM UTÁNZATOK

1987/5.


Találkozunk
a
tavaszi BNV-n
Keresse
FOTO.
ELEKTRONIK
emblémát!
IBM

5. ÉVFOLYAM
1987/5. SZÁM

μ mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Címképünk:
Ramocsai Imri munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Petrus György
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.
86—253

Szikra Lapnyomda
Budapest (87-0477)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Levél a kórházból 2
A zűrzavartól a zsibongásig 10
Távkonferencia II. 12
Adok-veszek-cserélek 14
μINFORM 15
RAINBOW a μM-ban 16
Mesterséges értelem 18
Vigyázat! Tolvaj! 27
A szoftvergyártás nagyjai 28
Olvastunk . . . 38

ISKOLA — SZÁMÍTÓGÉP

Élmény a növény: szimulációs modell a középiskolában 3
Szövegszerkesztő 5
Adatátadás BASIC-ből gépi kódba 7
OKTA-TOTÓ 8

DIÁKROVAT

Rajzolóprogram 9
Karakteres képernyőt másoló rutin 10

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód 23
Teknősbéka-grafika BASIC-ben 24
MERGE 26

μPROGRAMOK

Disassembler program 30
Különleges input 32

μKLUB

Printerszalag házilag 33
A lemezár rejtélye 34
Z80 szimulátor 35
Integrált szoftver 36

SAKKPROGRAMOZÁS

Bitek és figurák 42

JÁTÉKPROGRAMOK

CSIPP-CSEPP 43
Apróságok 44
Szellem ellen szellemesen 45

AZ OLVASÓ ÍRJA 45

KÖNYVEK 47

HÍREK-ÉRDEKESSÉGEK 48

Levél a kórházból

„. . . nálunk a legnagyobb visszahúzó erő: az emberekben élő konzervativizmus. Hiányzik a gondolkozásunkból épp úgy, mint a vitáinkból a több szempontú megfontolás, a másik nézeteinek a tiszteletben tartása, megfigyelése. Túl sok bennünk az indulat, a »magyar virtus«, és túl kevés a nyitottság. Mi büszkén elvetjük, megkövezzük, ami szokatlan. De ha más népek dicsérete hitelesíti, megbocsátunk. Esetleg még be is fogadjuk."
(Mészáros Márta — Molnár Mária interjúja, Képes 7—1987/5)

Február, jég, lábtörés, kórház. Tulajdonképpen magánügy, csak hirtelen időmilliomossá váltam, ami egy meglehetősen új, szokatlan állapot számomra. Nincs telefon, napi egyszeri alkalomra szűkült a kapcsolat a külvilággal. Ideális alkalom a szerkesztőségi cikk megírására, csodálkozik, majd a szerkesztőség, hogy sürgetés nélkül, időben megkapják az írást. Ráérek, mások dolgoznak velem.

Egy nagyon jó hangulatú kórterembe kerültem, voltaképpen csak két napja vagyok még itt, így én vagyok az új beteg. A többieket az elmúlt napokban operálták —, én előtte vagyok még a procedurának –, így ők már hazafelé készülnek. Beszélgetnek. Még nem nagyon tudok bekapcsolódni a társalgásba, hiszen nincsenek közös élményeink, néha kapok egy-egy kérdést, kádereznek.

Én is figyelem őket, arra vagyok kíváncsi, hogy előjön-e egyszer is az informatika, kibuknak-e valahogyan itt is azok a — véleményem szerint — társadalmi méretű problémák, amelyek az elmúlt időben életem fontos részévé váltak, és amelyeknek megoldására szeretném a Neumann Társaság tagjait (és mindazokat, akik velem együtt úgy látják, hogy társadalmi program nékül nincs elektronizálási program sem) megnyerni. Hiszen, hiába fejlesztünk ki csodálatos számítógépeket, európai szintű vagy világhírű programokat, ha nem teremtjük meg ehhez az *érdeklődő* és *igényes* társadalmi közeget! Sikerre, gazdasági előrehaladásra enékül nem számíthatunk.

Teljesen világos, hogy egy kórházi kórterem eleve alkalmatlan a társadalom „informatikai erővonalainak" vizsgálatára, hiszen itt mindenki a saját nyomorúságával van elfoglalva. E környezetben csak az abszolút véletlenre számíthattam, de ez sem „jött be." Nem esett szó az informatikáról . . .

Sajnos a munkahelyeken sem sokkal kedvezőbb a helyzet. Elnézést kérek a hasonlatért, de az embernek az az érzése, mintha a magyar gazdaság is egy nagy kórterem lenne, ahol mindenki a saját napi problémáival van elfoglalva, és ezért nincs ideje egy kicsit előretekinteni, időt és energiát fordítani arra, hogy a szűkebb vagy tágabb környezetében jobban menjenek a dolgok.

Nagyon sok embernek elmondtam az elmúlt hónapokban — amikor a Teleteaching '86 után igen megélénkült az érdeklődés a távtanulás és egy Nyitott Egyetemi képzés megindítása iránt — hogy fel kellene már ébredni Csipkerózsika álmunkból, és a szekerünk rúdját abba az egyáltalán nem kényelmes és nem is mindig kellemes utat ígérő irányba kellene fordítanunk, amelyet a folyamatos megújulási készség, az új ismeretek megszerzésére való lankadatlan törekvés, egyszóval a folyamatos továbbtanulás jellemez.

A HVG-ben cikket váltottunk Tömpe Zoltánnal a hazai szoftverhelyzetről (első vérig, bandázs nélkül). Én azt hiszem – pedig azóta nem sok hónap telt még el –, hogy ma azt a cikket már másként kellene mindkettőnknek megírni, mert e néhány hónap alatt is nagyon sokat változott az informatikai világ. Még a század közepén is — szerte a világon — az volt a divat, hogy az ember megszerezte a diplomát és ennek birtokában az élete végéig mérnök úr, doktor úr, ügyvéd úr volt nemcsak a megszólítása, de a társadalmi pozíciója is. Az egyszer megszerzett tudás az évek során gazdagodó tapasztalattal elegendő volt arra, hogy az ember a választott munkakörét élete végéig ellássa.

Ha az ember az iparilag fejlett és főleg az informatikában vezető helyen álló országok szakembereivel beszélget, akkor kiderül, hogy ezekben az országokban a helyzet az elmúlt néhány évben nagyon megváltozott. Az „öreg" diplomák folyamatosan vesztik az értéküket, a napról napra megjelenő újdonságokat meg kell tanulni, és egyáltalán nem elég az a tapasztalat, amit az ember a munka során összegyűjt. Ezért azután rendkívüli mértékben megnőtt az érdeklődés mindenféle szintű tanfolyam és továbbképzés iránt. Bonyolítja a helyzetet és sürgeti a megoldást, hogy a gazdaság modernizálása napok alatt eltüntet akár egész iparágakat is, és akik nem gondoskodnak időben arról, hogy új és „kapós" munkájuk legyen, azok egykettőre munkanélkülivé válnak. Nagyon jól tudom én is, hogy kegyetlen dolgokról van szó, de a műszaki fejlődésnek napjainkban tapasztalt felgyorsulása — akaratunktól függetlenül — ilyen gazdasági döntések sorozatát eredményezi.

A társadalomnak — a miénknek is — ezekre a változásokra fel kell készülnie. Lehet, hogy túlságosan leegyszerűsítem a problémát — egy előadásom utáni vitában az egyik hozzászóló is ezt mondta —, ennek ellenére én úgy látom, hogy a legfontosabb kérdés ma nálunk is a társadalmi méretű alap- és továbbképzés feltételeinek a megteremtése. Nekünk is — mint másoknak is — a technikailag fejlettebb társadalomba csak az oktatási alapok lerakása után van esélyünk továbblépni. Igaz — válaszolnék kérdezés nélkül is az olvasónak —, ebben is, mint annyi másban, évtizedekkel el vagyunk maradva, de mint annyi másban is, ebben is az elmaradásunkat — nyilvánvalóan nem kevés beruházással — előnnyé is változtathatnánk.

Az most a feladatunk — ez a leghatározottabb véleményem —, hogy a távtanulást, a nyílt egyetemi oktatást a lehető legrövidebb idő alatt meg kell szerveznünk. Ehhez persze nem kell a távtanulás speciális hazai technológiáját kitalálnunk, hanem át kell vennünk onnan — pl. a Brit Open University-től —, ahol ezt már évtizede sikeresen alkalmazzák. Össze kell fognunk — erre az NJSZT nem csak kész, de a VII. ötéves tervben feladatul is kapta az elektronika oktatásának lakossági részprogramját, illetve ennek megszervezését — a meglévő vagy éppen kezdődő távtanulási kezdeményezéseket az egyetemeken, vezetőképző intézetekben, különböző tanfolyamok keretében és a nyelviskolákon. Egy nagyon felületes felmérésemből is kiderült, hogy megfelelő koordináció hiányában sajnos máris párhuzamosan futnak bizonyos tevékenységek a tananyag összeállítására és kidolgozására, de nem kell sok idő ahhoz sem, hogy ugyanez történjék a beszámoltatási és vizsgáztatási rendszerekben is.

Bossányi Katalin kitűnő cikket írt a Népszabadságban, miszerint kevés nálunk a gazdasági társulás, pedig ahhoz, hogy egy gazdaság eredményes legyen, össze kell fogni a meglévő erőforrásokat — például társulások, társaságok formájában. Hozzáteszem, hogy ezeknek azután folyamatos megújulásra készen, rugalmasan kell működniük, és működtetniük is a tagok erőforrásait. Továbbá remélem, hogy — bár biztosan nagyon sok tárgyalás után — az oktatás felelősei és munkásai ebben példát fognak mutatni, és időben sikerül egy széles körű informatikai távtanulási rendszer beindítása.

KOVÁCS GYŐZŐ

Nem csak a biológia fakultásnak

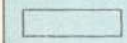
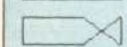
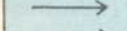
Élmény a növény: szimulációs modell a középiskolában

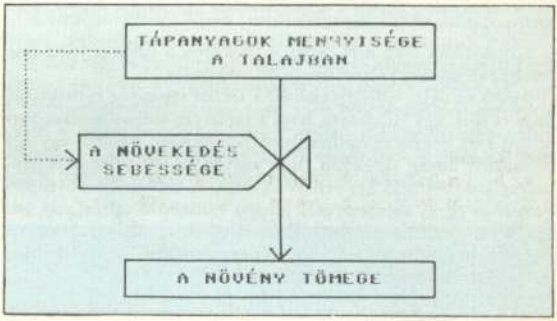
Szeretném megmutatni, hogyan lehet a biológiában eddig mostohán kezelt területeken — ahol a matematikai analógia nem annyira nyilvánvaló — hasznositani a számítógépeket az ismeretek közvetitésében. Nem ördöngősség ugyanis sem az egyszerű szimulációs modellek készítése, sem az, hogyan kell ezekből programokat írni, majd a tanulás segítésére felhasználni. Az *1. ábra* maga is tükrözi, hogy milyen csekély és mennyire áttekinthető fogalommal, illetőleg rendszerrel dolgozhatunk.

A modellek a középiskolás tananyaghoz kapcsolódnak, a modellezendő jelenségek a tananyag részei. Sem elkészítésükhöz, sem használatukhoz a tananyagot meghaladó ismeret nem szükséges, hiszen mindössze arról van szó, hogy próbáljuk meg a tankönyv leíró kijelentéseit életszerűvé tenni.

Tulajdonképpen nem is ez a néhány modell a fontos, hanem a módszer érzékeltetése, aminek példáján számtalan hasonló szemléltetés születhet tanárok és diákok keze alatt.

1. ábra. A modellezés jelölései

JELÖLÉS	JELENTÉS
	SZINT TIPUSÚ VÁLTOZÓ
	SEBESSÉG TIPUSÚ VÁLTOZÓ
	KÜLSŐ PARAMÉTER
	ANYAG ÁRAMLÁS
	INFORMÁCIÓ ÁRAMLÁS



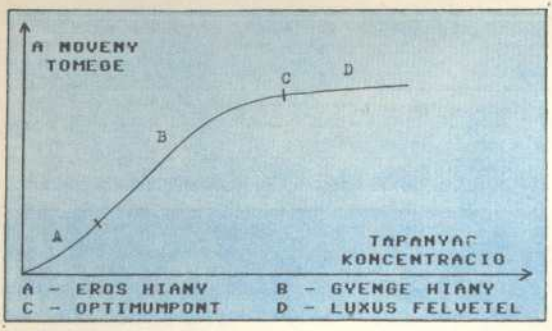
2. ábra. A növény növekedésének függése a talaj ásványianyag (tápanyag) koncentrációjától

A modellezés alapja

A gimnáziumok III. osztályosainak tankönyvében találhatjuk többek között a növények életjelenségeivel foglalkozó fejezeteket. Ebből a témakörből választottam ki két jelenséget — a növények tápanyag- és vízfelvételét —, és ezeket modelleztem.

A modellek természetesen erősen idealizáltak lesznek: nem egy konkrét növény viselkedését akarom leírni, hanem egy általánosan érvényes sémát szeretnék mutatni, mely igazából egyetlen növényt sem ábrázol, de lényegében majd mindegyikre érvényes.

Mindenki számára nyilvánvaló, hogy a jobban táplált növény szebben, gyorsabban nő, több termést hoz. Ennek a jelenségnek készítsük el egy egyszerű modelljét!



3. ábra. A növények növekedésének függése az ásványi tápanyag mennyiségétől

A növények ásványianyag-felvétele

A *2. ábra* a következőt jelenti: a növény tömegének növekedése csak a talajban lévő tápanyagok mennyiségétől függ, és ez a hatás a növekedés sebességének befolyásolásán keresztül érvényesül.

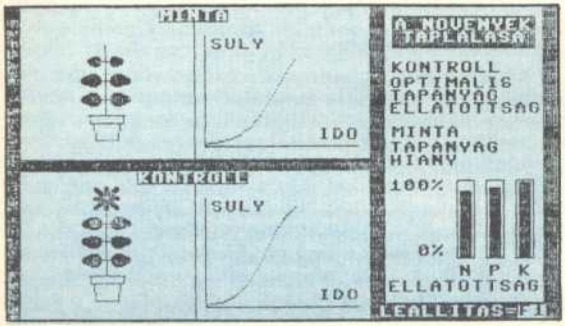
Különböző növényeken végzett vizsgálatok azt mutatták, hogy a növekedés a legfontosabb ásványi anyagok mennyiségétől általában a *3. ábrán* látható módon függ. Ez egy ún. tapasztalati görbe.

Mindezek ismeretében, de azzal az egyszerűsítéssel, hogy csak az optimumig terjedő szakaszt szimuláljuk, alkossuk meg a modellt. A különböző természettudományos jelenségek közötti analógiákra asszociálva, ezen a ponton sokszor segíthetünk magunkon. Most is, például valószínűleg nem követünk el nagy hibát, ha a tapasztalati görbét a szimulációban egy telítési függvénnyel közelítjük.

Gondoljunk csak arra, hogy a fizikában a vasmag telítődése valójában szintén valami hasonló „befogadás."

Mivel az egyes növényfajoknál a felvett tápanyag mennyisége más és más, úgy gondolom, hogy a tápanyag-mennyiségeket normálni kell, vagyis az optimumponthoz minden elemből egy-egy egységnyi anyag tartozzon (más szóval az optimumnál a tápanyag-koncentráció egy relatív egység). A különböző növényfajoknál az egyes komponensek hiánya különböző súllyal esik latba; a mi általános modellünkben mindhárom tápanyag legyen egyformán fontos. Ez már durvább egyszerűsítés, de ne feledjük, hogy csak absztrakt modellt alkotunk!

4. ábra. A modell felhasználásával készült program egyik képernyőállapota



Tehát a növekedés egyenlete öltse a következő alakot:

$$dN/dt = K.N\ (1 - N/Nmax)$$

ahol N a növény tömege, Nmax az optimális táplálással elérhető maximális tömeg, K a növekedés sebességi állandója, t az idő.

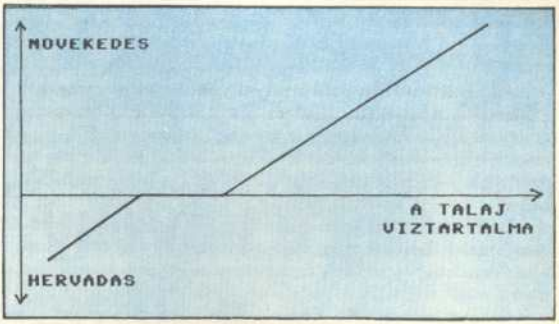
Természetesen a differenciálegyenletre nincs szükségünk, ezért átalakítjuk könnyebben kezelhetővé:

$$N = K.N\ (1 - N/Nmax).\Delta t$$

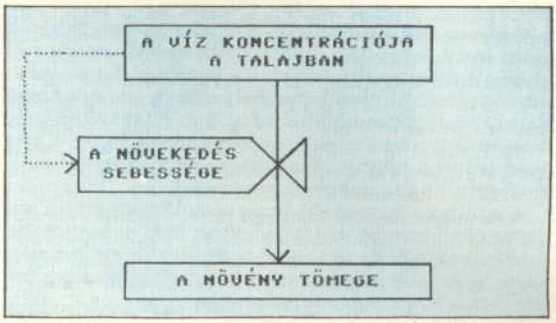
A sebességi állandót pedig számoljuk a következő módon:

$$K = \sqrt{\frac{n.p.k}{100}}$$

ahol n, p és k a nitrogén, foszfor és kálium koncentrációja (relatív egység) a talajban.

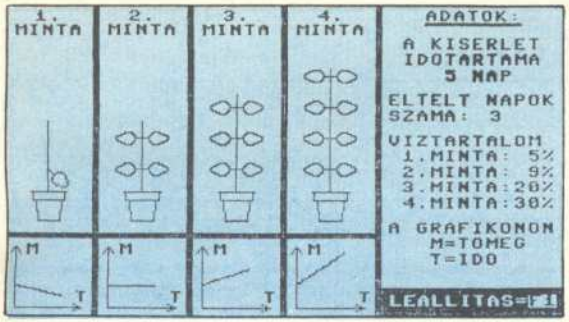


5. ábra. A növény sorsa a vízfelvétel függvényében



6. ábra. A növény növekedésének függése a talaj víztartalmától

Ezzel el is készült a modell. Most már nincs más hátra, mint azt a programot megírni, mely ennek az igen egyszerű egyenletnek a felhasználásával bemutatja a növények növekedését különböző tápanyag-ellátottság esetén. A program kvalitásait nyilvánvalóan behatárolja a számítógép, amelyen készül; mások a lehetőségek egy jó grafikájú gépen, és megint mások egy gyengébben. Kedvcsinálóként bemutatok egy képernyőképet ennek a modellnek C64-es gépen futó programjából *(4. ábra)*. A képen látható két növény egyikének tápanyag-ellátottsága optimális (ez a kontroll), a másik pedig a kísérleti alany (minta). Mindkét növény növekedését figyelemmel kísérhetjük grafikonon, de érzékelhetjük a növényeken a morfológiai változásokat is: például a kontroll már kivirágzott. A kép jobb oldalán az oszlopdiagram mutatja, hogy milyen tápanyag-koncentráció okozza ezt a hatást.



7. ábra. A modell felhasználásával készült program egyik képernyőállapota

A növények vízfelvétele

Ez a modell sokban hasonlít az előzőhöz. A különbség, hogy itt nemcsak a növekedés, hanem ennek ellentéte, a hervadás is számításba van véve. Tudjuk, hogy a talajban lévő víz a növények szempontjából felvehető és fel nem vehető részre oszlik. A felvehető víz tartományában a növények növekedése és a víztartalom között az összefüggés az intervallum elején lineáris. A fel nem vehető víz tartományában újabb két részt lehet elkülöníteni: az elsőben a növény nem növekszik, de még nem is hervad (legalábbis rövid ideig), a másik a hervadási tartomány *(5. ábra)*.

A modellben a vízfelvételt befolyásoló külső tényezőket nem vesszük figyelembe, így a növekedés sebességére az egyetlen befolyással bíró tényező a talaj víztartalma *(6. ábra)*.

Ezek után lássuk a mechanizmust. A felvehető víz tartományában a mérések szerint

$dN/dt = konstans$

de a konstans számértékének nagysága függ a vízkoncentrációtól. Ha a növény olyan víztartalmú talajban van, hogy se nem növekszik, se nem hervad, akkor

$dN/dt = 0$

Ha pedig a hervadási zónában van, akkor

$-dN/dt = konstans$

Ezek alapján megalkothatjuk egyszerű modellünket a következő módon:

$\Delta N = konstans . \Delta t$

ahol a konstans a felvehető víz tartományában pozitív, a hervadási tartományban negatív, a kettő között nulla, értéke pedig lineáris függvénye a talaj víztartalmának. Ehhez a példához is bemutatok egy képernyőképet *(7. ábra)* a modellt megvalósító C64-es programból.

A modell viselkedése leolvasható a képekről, grafikonokról. Ez és még inkább sok más, egyéb módon nehezen vagy egyáltalán nem vizsgálható jelenség a folyamatában érzékeltetve percek alatt többet tanít meg, mint esetleg órák múltán a szárazabb leírások. Egy-egy tanórai demonstráció előre kipróbált, jellegzetes paraméterekkel futtatva, esetleg néhány bemutató kísérlet után rátérve a rendszer viselkedésének megfejtésére, igazi lázba hozza a fantáziát: következtetni lehet a kiindulási helyzetre, és viszont. Ha nem csak grafikus megjelenítést alkalmazunk, akkor a numerikus adatokkal szinte tetszőleges kísérletsorozatokat is végeztethetünk, ha pedig a modellt kibővítjük elméleti ismereteket és számonkérő részt is tartalmazó oktatóprogrammá, akkor az önálló tanuláshoz is alkalmas lesz.

DEMETER LÁSZLÓ

HT-1080Z

Szövegszerkesztő

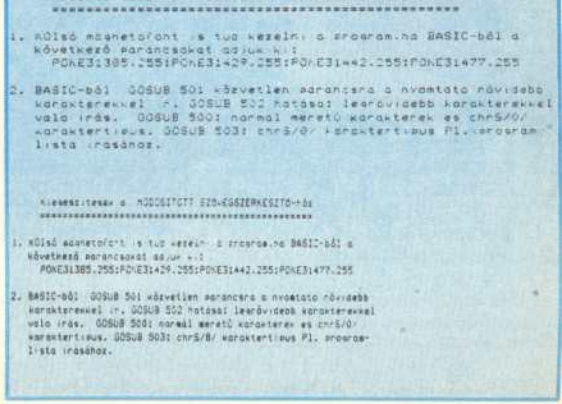
Kezdjük a rosszal! A program kissé távol áll attól, amit egy komoly szövegszerkesztő teljesít — nem ismeri az összes ékezetes betűt, csak egyetlen képernyőnyi információt lehet vele egyszerre kezelni: a képernyő tartalmát magnóra felvenni, betölteni vagy nyomtatóra küldeni. Továbbá: a vezérlés elfoglal néhány írásjelet. A nyomtató típusa ROMOM RM 6311. A beszúrás úgy, hogy a többi karakter jobbra eltolódjon nem megy, de erre az aránylag kis képernyőterület esetében nincs is sok szükség, és a kezelés további billentyűket igényelne.

Folytassuk a jóval! Így is elég hatékony a szerkesztés, kis gyakorlással jól lehet mozogni a shiftelt nyílbillentyűkkel a képernyőn. Hiányosságai ellenére sokat segít a program; továbbfejlesztheti, aki igényesebb és jobbak a lehetőségei. A képernyőre írás és nyomtatás mellett programlista nyomtatására is alkalmazható.

Az RM 6311-es nyomtatóval ugyanis az a probléma, hogy a LLIST parancs az A4-es papíron végig ír, és kisebb szélességű papírt nem fogad el. Ezzel a programmal — főként, ha a nyomtató karakterméretét is előre beállítjuk — ha a képernyőre tetszőleges program listáját vagy annak egy részletét kiíratjuk, POKE16526,74:POKE16527,122:X=USR(Ø) parancsokkal behívhatjuk a szövegszerkesztő gépi részét, majd a képet módosítva, a felesleges részeket letörölve, a dupla kereszt jellel a nyomtatást elindíthatjuk. Természetesen ez esetben a szövegszerkesztő program BASIC részét előzőleg legalább egyszer futtattuk, majd a BASIC részt törölve, helyébe az adott programot töltöttük be.

A gépi rutinokból RESET gombbal vagy az „&" jellel ki lehet jönni. Fontos, hogy a gép bekapcsolása után a RAMTOP 31232-re legyen beállítva. A memóriacímek átírásával módosított szövegszerkesztő külső magnóval is dolgozhat. (1. ábra)

A program segítségével a képernyőn tetszőleges szöveget írhatunk, a sorok és betűk között a shiftelt nyílbillentyűkkel lehet mozogni, a képernyőképet magnóra küldhetjük, a magnóról másokat behívhatunk és a nyomtatón kiírathatjuk.



```
          SZÖVEGSZERKESZTÖ  JZ
10 DATA 62,14,205,51,0,195,71,122,205,201,1,205,41,122,201,62,1,
50,156,64,62,27,205,42,3,62,91,205,42,3,62,57,205,42,3,62,96,205
,42,3,201,33,0,60,6
20 DATA 64,62,129,119,35,16,252,17,63,0,6,14,43,35,119,25,119,16
,250,6,64,119,35,16,252,201,205,8,122,62,14,205,51,0,205,43,0,40
,251,254,38,200,254,35,40
30 DATA 18,254,36,40,57,254,37,40,101,254,39,204,8,122,205,51,0,
24,226,62,15,205,51,0,62,1,50,156,64,33,0,60,6,16,14,64,62,10,20
5,42,3,205,15,122,126
40 DATA 205,42,3,35,13,32,248,16,236,62,14,205,51,0,195,79,122,6
2,0,205,18,2,62,15,205,51,0,1,48,117,205,96,0,205,135,2,1,255,3,
33,0,60,126,205,100



50 DATA 2,35,11,121,176,32,246,62,0,205,248,1,62,14,205,51,0,195
,79,122,62,0,205,18,2,205,150,2,33,0,60,1,255,3,205,53,2,119,35,
11,121,176,32,246,62
60 DATA 0,205,248,1,62,0,50,34,64,62,14,205,51,0,195,79,122
100 CLS:PRINT"      * * * * * * * * * * * * * * * * * * * * * * *
* * * * *"
120 PRINT"      *     SZOVEG-SZERKESZTO  HT-1080Z - ROMOM 6311
 JZ    *"
130 PRINT"      * * * * * * * * * * * * * * * * * * * * * * *
* * * *":PRINT:PRINT"A PROGRAM HASZNALATA:":PRINTSTRING$(21,"=")
:PRINT
130 PRINT"      * * * * * * * * * * * * * * * * * * * * * * *
* * * *":PRINT:PRINT"A PROGRAM HASZNALATA:":PRINTSTRING$(21,"=")
:PRINT
140 PRINT"  - KURZOR MOZGATASA ..... SHIFT + NYILBILLENTYUK"

150 PRINT"  - KEP TORLESE ..... ' /FELSO VESSZO/"
160 PRINT"  - FELVETEL MAGNORA ..... $ /DOLLARJEL/"
170 PRINT"  - BETOLTES MAGNOROL ..... % /SZAZALEKJEL/"
180 PRINT"  - NYOMTATAS ..... # /DUPLA KERESZT/"
190 PRINT:PRINT:PRINT"HA ELOLVASTAD,NYOMJ MEG EGY BILLENTYUT!"
200 A$=INKEY$:IFA$=""THEN200
210 CLS:PRINT@64+20,"GEPI KOD BETOLTESE FOLYIK!":FORI=31232TO3
1473:READY:POKEI,Y:NEXT:CLS

220 POKE16526,0:POKE16527,122:X=USR(0):LPRINT
230 CLS:PRINT"HA A PROGRAMOT ARRA AKAROD HASZNALNI,HOGY EGY KEPE
RNYO TARTAL-":PRINT"HAT NYOMTATORA VIDD:  'K' BILLENTYU":PRINT"U
TANA A PROGRAM BREAK-KEL MEGALL.
232 PRINT"A KEPERNYO TELEIRASA UTAN  BE KELL IRNI AZ  X=USR(0)":
PRINT"PARANCSOT, MAJD MEGNYOMNI A '#' BILLENTYUT - A SZOVEG A NY
OMTA-":PRINT"TON KIIRODIK."
240 PRINT:PRINT"HA UJBOL SZOVEG-SZERKESZTEST AKARSZ:  'S' BILLEN
TYU"
245 PRINT:PRINT"U J R A    I N D I T A S:    RUN":PRINT"    /HA
FUT A PROGRAM,AKKOR ELOTTE: BREAK/"
250 A$=INKEY$:IFA$=""THEN250ELSEIFA$="K"THENPOKE16526,74:POKE165
27,122:STOPELSEIFA$="S"THEN220ELSE250
```

A ROMOM RM 6311 nyomtató a grafikus karaktereket alapüzemmódban nem tudja kivinni, ezért ha a képernyőn ilyen van (például keret), az a magnószalagon rögzítődik ugyan, de a papíron üres lesz a helye.

Hogy az ékezetes betűk közül legalább a fontosabbakat ne nélkülözzük, a nyomtatót LPRINT CHR$ (127) paranccsal érdemes alaphelyzetbe állítani, így a relációs jelek helyett ő betűt, a zárójelek helyett ú és ű betűt kapunk. A szöveg ezzel már tűrhetően olvasható.

A vezérlő karakterek: a dupla kereszt, a dollárjel, a százalékjel, az et-jel és a felsővessző. Ezeket íráskor figyelmen kívül kell hagynunk.

Munka a programmal

— KURZOR MOZGATASA . SHIFT + NYILBILLENTYUK
— KEP TORLESE ..... , (FELSOVESSZO)
— FELVETEL MAGNORA ..... $ (DOLLARJEL)
— BETOLTES MAGNOROL ..... % (SZAZALEKJEL)
— NYOMTATAS ..... # (DUPLA KERESZT)
HA ELOLVASTAD, NYOMJ MEG EGY BILLENTYUT!

Ez a kép jelenik meg a képernyőn, ha RUN paranccsal elindítjuk a programot. A vezérlő karakterek leolvashatók.

Bármelyik gomb megnyomása után néhány másodperc alatt a DATA sorokból betöltődik a gépi kód, majd a program elindul, törli a képet, és egy keretet készít. A papíron a nyomtató ezt nem

```
                ORG  31232  ; * * *  SZOV SZERK  * * *
                LOAD 31232
 7A00 3E0E      LD   A,14
 7A02 CD3300    CALL 33H    ; CURSOR ON
 7A05 C3477A    JP   KEZD
                ;------------------------------
 7A08 CDC901    CLS:  CALL 1C9H   ; KEP TORLES
 7A0B CD297A           CALL KERET
 7A0E C9               RET
                ;------------------------------
 7A0F 3E01      POS:  LD   A,1    ; MARGO BEALLITAS
 7A11 329C40           LD   (409CH),A:KIMENETJELZO
 7A14 3E1B             LD   A,27
 7A16 CD2A03           CALL 32AH   ; OUTCHR
 7A19 3E5B             LD   A,91
 7A1B CD2A03           CALL 32AH
 7A1E 3E39             LD   A,57   ; MARGO
 7A20 CD2A03           CALL 32AH
 7A23 3E60             LD   A,96
 7A25 CD2A03           CALL 32AH
 7A28 C9               RET
                ;------------------------------
 7A29 21003C    KERET: LD   HL,3C00H; KERETEZES
 7A2C 0640             LD   B,64
 7A2E 3E81             LD   A,129
 7A30 77        LB1:   LD   (HL),A
 7A31 23                INC  HL
 7A32 10FC              DJNZ LB1
 7A34 113F00            LD   DE,63
 7A37 060E              LD   B,14
 7A39 2B                DEC  HL
 7A3A 23        LB2:   INC  HL
 7A3B 77                LD   (HL),A
 7A3C 19                ADD  HL,DE
 7A3D 77                LD   (HL),A
 7A3E 10FA              DJNZ LB2
 7A40 0640              LD   B,64
 7A42 77        LB3:   LD   (HL),A
 7A43 23                INC  HL
 7A44 10FC              DJNZ LB3
 7A46 C9                RET
                ;------------------------------
 7A47 CD087A    KEZD:  CALL CLS
 7A4A 3E0E              LD   A,14
 7A4C CD3300            CALL 33H    ; CURSOR ON
 7A4F CD2B00    BILL:  CALL 2BH    ; BILL FIGY
 7A52 28FB              JR   Z,BILL
 7A54 FE26              CP   '&'
 7A56 C8                RET  Z      ;   BASIC SORBA VISSZA
 7A57 FE23              CP   '#'
 7A59 2812              JR   Z,KERESZT: NYOMTATORA
 7A5B FE24              CP   '$'
 7A5D 2839              JR   Z,DOLLAR: MAGNO FELV
 7A5F FE25              CP   '%'
 7A61 2865              JR   Z,SZAZ : BETOLTES MAGNOROL
 7A63 FE27              CP   39
 7A65 CC087A            CALL Z,CLS  ; KEP TORLES
 7A68 CD3300            CALL 33H
 7A6B 18E2              JR   BILL
                ;------------------------------
```

jeleníti meg; nyugodtan felülírható. Azért kell csupán, hogy lássuk a képmező szélső karaktereit. A kurzor a shiftelt nyilbillentyűkkel minden irányban mozog, de ha NEWLINE-nal térünk át a következő sorba, és éppen a legalsó sorban voltunk, a kép feljebb ugrik. Ezt kerüljük el! Ugyanígy a képernyő legutolsó (jobb alsó sarok) helyére semmi nem írható! Visszatörléshez a visszafelé nyíl használatos, a SPACE billentyű előre töröl. A shiftelt nyilbillentyűkkel ki lehet menni a képernyő szélein, a túlsó oldalon visszajön a kurzor.

Fontos! A HT—1080Z iskolaszámítógép bekapcsolása után a READY?-re 31232-t írjunk be. Utána NEWLINE. Így a gépi rutin a RAMTOP fölé, védett helyre kerül. Ezután kapcsoljuk be a SYSTEM-bővítést: SYSTEM NEWLINE, majd *?-re /12299 és NEWLINE. Ezután tölthetjük be a programot.

Alkalmazási tudnivalók

1. Mielőtt a programot elindítjuk, állítsuk be a nyomtató megfelelő karakterkészletét.

2. Menet közben a programot a RESET gombbal szakíthatjuk meg.

3. Újraindításhoz nem szükséges a program elejét végigvárni, GOTO 220 is jó.

```
  61 7A6D 3E0F    KERESZT:LD   A,15    : NYOMTAT
  62 7A6F CD3300          CALL 33H     : CURSOR OUT
  63 7A72 3E01            LD   A,1
  64 7A74 329C40          LD   (409CH),A: KIMENET NYOMT
  65 7A77 21003C          LD   HL,3C00H: KEPERNYO ELEJE
  66 7A7A 0610            LD   B,16    : SOROK SZAMA
  67 7A7C 0E40    L4:     LD   C,64    : KARAKTEREK SZAMA
  68 7A7E 3E0A            LD   A,10
  69 7A80 CD2A03          CALL 32AH    : SOREM
  70 7A83 CD0F7A          CALL POS     : POZICIONALAS
  71 7A86 7E      L3:     LD   A,(HL)
  72 7A87 CD2A03          CALL 32AH    : KAR. NYOMT.
  73 7A8A 23              INC  HL      : KOV KAR.
  74 7A8B 0D              DEC  C
  75 7A8C 20F8            JR   NZ,L3
  76 7A8E 10EC            DJNZ L4
  77 7A90 3E0E            LD   A,14    : CURSOR ON
  78 7A92 CD3300          CALL 33H
  79 7A95 C34F7A          JP   BILL
  80              ;-------------------------------
  81 7A98 3E00    DOLLAR: LD   A,0     : MAGNO FELV.
  82 7A9A CD1202          CALL 212H    : MOTOR BE
  83 7A9D 3E0F            LD   A,15
  84 7A9F CD3300          CALL 33H     : CURSOR OUT
  85 7AA2 013075          LD   BC,30000
  86 7AA5 CD6000          CALL 60H     : VARAKDZIK
  87 7AA8 CD8702          CALL 287H    : WRHDR
  88 7AAB 01FF03          LD   BC,1023 : SZAMLALO
  89 7AAE 21003C          LD   HL,3C00H
  90 7AB1 7E      L1:     LD   A,(HL)
  91 7AB2 CD6A02          CALL 26AH    : WRBYTE
  92 7AB5 23              INC  HL
  93 7AB6 0B              DEC  BC
  94 7AB7 79              LD   A,C
  95 7AB8 B0              OR   B
  96 7AB9 20F6            JR   NZ,L1
  97 7ABB 3E00            LD   A,0     : BELSO MAGNO
  98 7ABD CDF801          CALL 1F8H    : CASOFF
  99 7AC0 3E0E            LD   A,14
 100 7AC2 CD3300          CALL 33H     : CURSOR ON
 101 7AC5 C34F7A          JP   BILL
 102              ;-------------------------------
 103 7AC8 3E00    SZAZ:   LD   A,0     : BETOLTES MAGNOROL
 104 7ACA CD1202          CALL 212H    :DEFCAS
 105 7ACD CD9602          CALL 296H    : RDHDR
 106 7AD0 21003C          LD   HL,3C00H
 107 7AD3 01FF03          LD   BC,1023
 108 7AD6 CD3502  L2:     CALL 235H    :RDBYTE
 109 7AD9 77              LD   (HL),A
 110 7ADA 23              INC  HL
 111 7ADB 0B              DEC  BC
 112 7ADC 79              LD   A,C
 113 7ADD B0              OR   B
 114 7ADE 20F6            JR   NZ,L2
 115 7AE0 3E00            LD   A,0     : BELSO MAGNO
 116 7AE2 CDF801          CALL 01F8H   : CASOFF
 117 7AE5 3E00            LD   A,0     : CURSOR ELNYOMAS
 118 7AE7 322240          LD   (4022H),A
 119 7AEA 3E0E            LD   A,14
 120 7AEC CD3300          CALL 33H     : CURSOR ON
 121 7AEF C34F7A          JP   BILL
 122                      END
```

4. Szerkesztő üzemmódból úgy is kiléphetünk (míg a BASIC betöltő jelen van!), hogy a shiftelt 6-os billentyűt megnyomjuk. Képernyő nyomtatása után következő kilépéskor a program megáll, újra kell indítani.

5. Ha egyidejűleg más programmal is akarunk dolgozni, akkor NEW paranccsal (ha már a gépi kódok betöltése előbb megtörtént) a BASIC részt töröljük.

Ekkor a képszerkesztés indítása:
POKE 16526,Ø:POKE 16527,122:X=USR(Ø)

6. Bármilyen képernyő tartalmát (grafikus karakterek nélkül) — például egy másik program képernyőkimenetét vagy éppen egy rövid BASIC program listáját — így lehet nyomtatóra vinni:
POKE 16526,74:POKE 16527,122:X=USR(Ø)

Ez az utasítás BASIC programból is kiadható. A felesleges részeket a képről le is lehet törölni, esetleg hozzáírunk valamit, a dupla kereszt billentyű elindítja a nyomtatót.

7. Javításkor ne NEWLINE-nal álljunk a következő sorba, mert az törlődik. (A NEWLINE sortörlésre való!) Helyette a shiftelt lefelé nyíl ajánlható.

8. Vigyázat! A felvett kép gondosan letörölt, jó minőségű szalagra kerüljön! A visszaolvasó rutin mindent bevisz, amit a szalagon talál!

JUHÁSZ ZOLTÁN
tanár

PRIMO

Adatátadás BASIC-ből gépi kódba

A Primo gépi kódban írt programrészei és a BASIC program közötti adatátadással lapunkban már foglalkoztunk. Más utasítások alkalmazásával a gépi kódban megoldható feladatok köre tovább bővül.

A G7 jelű programban *(1. lista)* a képernyő egy pontját kapcsoljuk be, illetve ki. A szükséges adatokat a 170-es sorban adjuk át. A D változó és esetünkben a DE regiszter tartalmazza azt a memóriacímet, amelybe az első adatot elhelyeztük.

A gépi kódú résznél a DE regiszterben lévő adatot az úgynevezett verembe rakjuk a PUSH DE utasítással, majd átvesszük az IY regiszterbe a POP IY utasítással. Az IY regiszterhez hozzáadva a címzési eltolás értékét, hozzáférünk a kívánt adathoz (esetünkben az

1. lista

```
1 REM G7
100 C=0 : D=0 : E=0 : F=0 : X=0 : Y=0
105 CLS
110 DIM K%(20),L%(20)
120 C=VARPTR(K%(0))
122 D=VARPTR(L%(0))
140 POKEC,213,253,225,253,126,0,253,86,
      1,253,94,2,254,1,32,4,205,134,0,
      201,205,131,0,201
150 CLS
151 PRINT "0 RAJZOL"
152 PRINT "1 TOROL"
155 INPUT F
157 INPUT"X=";X
159 INPUT"Y=";Y
170 POKED,F,X,Y
180 E=CALL(C,D)
200 GOTO 151
```

```
-------------------------------
 A PROGRAM      .     A KOD
-------------------------------
PUSH DE         . 213
POP IY          . 253,225
LD A,(IY+0)     . 253,126,0
LD D,(IY+1)     . 253,86,1
LD E,(IY+2)     . 253,94,2
CP,1            . 254,1
JR NZ,4         . 32,4
CALL,134        . 205,134,0
RET             . 201
CALL,131        . 205,131,0
RET             . 201
-------------------------------
```

```
1 REM G8
100 C=0 : D=0 : E=0 : X=0 : Y=0
105 CLS
110 DIM K%(20),L%(20)
120 C=VARPTR(K%(0))
122 D=VARPTR(L%(0))
140 POKEC,213,253,225,253,86,1,253,94,
      2,205,137,0,40,5,6,1,253,112,0,
      201
150 CLS
155 INPUT "BEKAPCSOLJAM (I/N)";F$
157 INPUT"X=";X
159 INPUT"Y=";Y
170 POKED,0,X,Y
175 IF F$="I" THEN SET(X,Y)
180 E=CALL(C,D)
190 IF PEEK(D)=1 THEN PRINT"BEKAPCSOLT"
      ELSE PRINT"NEM BEKAPCSOLT"
200 GOTO 155
```

A PROGRAM	A KÓD
PUSH DE	213
POP IY	253,225
LD D,(IY+1)	253,86,1
LD E,(IY+2)	253,94,2
CALL,137	205,137,0
JR Z,5	40,5
LD B,1	6,1
LD(IY+0),B	253,112,0
RET	201

2. lista

```
1 REM G/IDOMERO
10 POKE16445,0,0,0
30 FOR I=1 TO 50 : NEXT I
40 T=PEEK(16445)+256*PEEK(16446)+
      65536*PEEK(16447)
70 PRINT T
```

3. lista

LD A, (IY+Ø), az LD D, (IY+1) és az LD E, (IY+2) utasításokkal).

A pont bekapcsolását a szoftverkönyvben leírtak szerint a 131., a kikapcsolását pedig a 134. címen kezdődő szubrutinnal végezzük. Az IY regiszter segítségével a gépi kódú részből a BASIC-be is adhatunk át adatokat. Ezt a G8 jelű program szemlélteti *(2. lista)*. A program a 137. címen kezdődő szubrutinnal megvizsgálja a képernyő egy adott pontját. Ha a pont be van kapcsolva, akkor a 170-es sorban „feltöltött" 3 db memóriarekeszből az elsőben lévő Ø értéket 1-re írja át. Ezt az értéket a 190-es sorban vizsgáljuk.

A gépi kódú programozásnál időnként célszerű megmérni, hogy az adott program mennyivel gyorsabb az azonos feladatú „tiszta" BASIC programnál. A G/IDOMERO program *(3. lista)* a gép órájának időmérési felhasználását mutatja be a 30-as sorbeli ciklus futási idejének mérésével.

Somogyi György

ÚTJÁRA INDUL

a hatrészes OKTA-TOTÓ. A kérdéseket minden alkalomra a számítástechnika meghatározott témaköréből válogattuk. Első ízben a számítástechnika történetéből.

A megfejtéseket a pályázati szelvényre kell rávezetni úgy, hogy a kérdésszám sorába beírja a játékban résztvevő kedves olvasó tippjét 1-est, 2-est ill. X-et. A három közölt válasz közül csak egy a helyes. Ezután a kitöltött szelvényt postai levelezőlapra felragasztva a szerkesztőség címére szívesenkérjék beküldeni. /1371 Budapest, Pf.: 433./

Beküldési határidő: 1987. május 24.

A helyes megoldás feladói közül öt nyertest sorsolunk ki havonta, akik 200 Ft értékű könyvutalványt kapnak. Ha valakinek mind a hat TOTÓ-ja telitalálatos, számítógépet nyer!net!
Pályázati szelvény nélkül érkezett megoldás kiesik a sorsolásból!
A rejtvényfejtéshez jó szórakozást kíván a TOTÓ összeállítója,

Mikes Gábor

1. Mi az Abakusz?
   1. A számítógépgyártásban használt szemcsés alapanyag, aminek belégzése kóhogésre ingerel.
   2. Már az ókorban is ismert számológép.
   X. Régi mértékegység: 1 Abakusz = 12 Fikusz.

2. Ki készített 1642-ben kiemelkedő jelentőségű számológépet?
   1. Descartes
   2. Kempelen Farkas
   X. Pascal

3. Mikor készítette el zsebszámológépét Leibniz, német matematikus és filozófus?
   1. 1620-ban
   2. 1673-ban
   X. 1750-ben

4. Ki használta a lyukkártyát elsőnek adatfeldolgozókónt statisztikai munkára?
   1. Herman Hollerith
   2. Charles Babbage
   X. Newton

5. Mi a rendezőgép?
   1. Számítógép által vezérelt mezőgazdasági segédeszköz.
   2. Lyukkártyák különböző szempontok szerinti sorba rakását végző gép.
   X. A modern szórótárakban a rendezést segítő számítógép.

6. Mit hívunk külső programozásnak?
   1. Programbevitelt a számítógéptől távolabb elhelyezett terminálról.
   2. A másodállásban - külső munkaerővel - elvégeztetett programozási munkát.
   X. Kapcsolótáblán, kapcsolózsinórokkal megvalósított programozást.

7. Az 1. generációs számítógép milyen áramköri elemet tartalmazott?
   1. Elektroncsövet
   2. Képcsövet
   X. Tranzisztort

8. Kinek a nevéhez fűződik a belső programozás elve?
   1. Kalmár László
   2. Neumann János
   X. Tarján Rezső

9. Mit jelent az ESZR rövidítés?
   1. Első Számítástechnikai Részvénytársaság
   2. Egységes Számítógépes Rendszer
   X. Elkészült Szubrutinok Rejtjelezése (titkosítás, másolás elleni védelem.)

10. Mikor szerezték be Magyarországon az első szervovezérléses robotot?
    1. 1961-ben
    2. 1972-ben
    X. 1985-ben

11. Melyik volt a Magyarországon gyártott legelső háziszámítógép?
    1. HT-1080Z
    2. PRIMO
    X. AIKOM-16

12. Melyik Magyarország első iskolaszámítógépe?
    1. HT-1080Z
    2. PRIMO
    X. VTC

13. Melyik cég vette meg a Sinclairt?
    1. COMMODORE
    2. IBM
    X. AMSTRAD

13+1. Melyik cég gyártmánya az AMIGA számítógép?
    1. COMMODORE
    2. IBM
    X. AMSTRAD

PÁLYÁZATI SZELVÉNY 87/5

OKTA-TOTO

Kérdés	Tipp
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
+	
13+1.	

Név: _____

Cím: _____

Rajzolóprogram

Először töltsük be a Beta-BASIC segédprogramot, és csak azután kezdjük el beírni a rajzolóprogramot. A program RUN-nal indul. Választanunk kell, hogy egy már előzőleg kimentett képet akarunk visszatölteni, vagy egy újat kezdünk el rajzolni. A kimentésnek két formája van: az egyik a SCREEN$ típusú, a másik a képet egy K$ nevű karakteres tömbbe eltárolva menti ki. Az így kimentett kép, ha Beta-BASIC-ben dolgozunk, egy POKE 16384, K$ utasítással elöhívható a képernyőre.

Új kép rajzolását a színek megkérdezése után kezdhetjük. A képernyő bal alsó sarkában a mindenkori pozíciónk X és Y koordinátája látható. Pozíciónkat a kurzorgombokkal változtathatjuk: ha Ø-t nyomunk, akkor az adott pozíción egy PLOT utasítást hajt végre a program. M betűt nyomva egy menüt kapunk eredményül további lehetőségeinkről:

DRAW	Egy vonalat húz a képernyőre a legutóbbi PLOT pozícióból a mostani pozíciónkhoz.
CIRCLE	A sugarat megkérdezve kört rajzol úgy, hogy a középpont a jelenlegi pozíció.
FILL	Ha egy zárt alakzaton belül állunk, akkor ezt INK színűre kifesti.
ATTRIBUTES	Új színeket adhatunk meg.
TÖRLÉS	Képünket törölhetjük a memóriából.
SAVE	Kazettára mentés SCREEN$ vagy karakteres tömb formában.

A program felépítése

1—3 értékek beállítása, kezdeti képek
1ØØ—315 programkezdő eljárások
12ØØ—1265 a fő eljárás (képrajzoló)
13ØØ—1355 a menü kiíratása
8ØØØ—8565 részprogramok
9ØØØ—9Ø55 betöltő eljárás
91ØØ—923Ø a menü kiíratását segítő eljárás

Ha a program bármilyen hibaüzenettel leáll, akkor: POKE 16384, K$:GOTO 123Ø és folytathatjuk a rajzolást.

UDVARI ANDRÁS

```
   1 PROC stringek
   2 PROC rajzolo
   3 PROC kiiratO
 100 DEF PROC rajzolo
 102 PAPER 7: CLS
 105 FOR i=Ø TO 31
 110 PRINT PAPER Ø;AT Ø,i;" ";AT
 21,i;"
 115 NEXT i
 120 FOR i=Ø TO 21
 125 PRINT PAPER Ø;AT i,Ø;" ";AT
 i,32;"
 130 NEXT i
 135 END PROC
 200 DEF PROC kiiratO
 205 PRINT INK 2;AT 4,9;"P A J Z
 O L O"
 210 PRINT INK 4;AT 17,5;"© 1986
 Udvari András"
 215 PRINT INK 1;AT 11,5;"Nyomj
 egy gombot!"
 220 GET a$
 225 FOR i=1 TO 40
 230 SCROLL 11,4,18,167,30,150
 235 NEXT i
 240 LET a$="     1.........KEP BE
TOLTESE"
 245 LET b$="     2.....UJ KEP RA
JZOLASA"
 250 FOR i=25 TO 1 STEP -1
 255 PRINT INK 3;AT 5,1;a$(i TO
 )
 260 PRINT INK 3;AT 14,1;b$(i TO
 )
 265 NEXT i
 270 LET a$=INKEY$
 275 IF a$="1" THEN GO TO 290
 280 IF a$="2" THEN GO TO 290
 285 GO TO 270
 290 FOR i=Ø TO 60
 295 SCROLL 12,4,8,167,30,160
 300 NEXT i
 305 END PROC
 310 IF a$="1" THEN PROC betolto
 315 IF a$="2" THEN PROC keprajz
olo
1200 DEF PROC keprajzolo
1205 INPUT "KERETSZIN:";b
1210 INPUT "PAPIRSZIN:";p
1215 INPUT "TINTASZIN:";i
1220 BORDER b: PAPER p: INK i: C
LS
1225 LET x0=Ø: LET y0=Ø
1226 PRINT AT 21,25;x0;AT 21,29;
y0
1230 LET a$=INKEY$
1235 IF a$="5" AND x0>Ø THEN LET
 x0=x0-1: PRINT PAPER p; INK i;A
T 21,25; USING "###";x0
1240 IF a$="6" AND y0>Ø THEN LET
 y0=y0-1: PRINT PAPER p; INK i;A
T 21,29; USING "###";y0
1245 IF a$="7" AND y0<175 THEN L
ET y0=y0+1: PRINT PAPER p; INK i
;AT 21,29; USING "###";y0
1250 IF a$="8" AND x0<255 THEN L
ET x0=x0+1: PRINT PAPER p; INK i
;AT 21,25; USING "###";x0
1252 IF a$="Ø" THEN PLOT INK i;x
0,y0: LET x1=x0: LET y1=y0
1255 IF a$="m" OR a$="M" THEN PR
OC menu
1260 GO TO 1230
1265 END PROC
1300 DEF PROC menu
1302 LET i=3
1305 LET k$=MEMORY$(16384 TO 1
6384+6912)
1315 FOR i=4 TO 14 STEP 2
1320 PRINT PAPER 7; INK Ø;AT i,4
;a$(i-1)
1325 LET i=i+1: NEXT i
1345 INPUT "Valasztasod:";v
1350 GO TO ON v;8000,8100,8200,8
300,8400,8500
1355 END PROC
8000 REM draw
8005 POKE 16384,k$
8010 LET x=x0-x1: LET y=y0-y1
8015 LET x1=x0: LET y1=y0
8020 DRAW INK i;x,y
8025 GO TO 1230
8030 REM END of draw
8035 REM ***
8100 REM circle
8105 INPUT "A kor sugara:";r
8110 POKE 16384,k$
8115 CIRCLE INK i;x0,y0,r
8120 GO TO 1230
8125 REM END of circle
8130 REM ***
8200 REM fill
8205 POKE 16384,k$
8210 FILL INK i;x0,y0
8215 GO TO 1230
8220 REM END of fill
8225 REM ***
8300 REM attributes
8305 INPUT "Keretszin:";b
8310 BORDER b
8315 INPUT "Papirszin:";p1
8320 INPUT "Tintaszin:";i
8325 POKE 16384,k$
8330 ALTER PAPER p TO PAPER p1
8335 LET p=p1
8340 GO TO 1230
8345 REM END of attributes
8350 REM ***
8400 REM clear
8405 INPUT "Tenyleg toroljek ? (
I/N)";b$
8410 IF b$="i" OR b$="I" THEN LE
T k$=""
8415 CLS : GO TO 1
8420 GO TO 1230
8425 REM END of clear
8430 REM ***
8500 REM save
8505 INPUT "SCREEN-kent vagy kar
akteres""tombkent mentsek ? (S/
K)";v$
8510 INPUT "Nev:";n$
8515 IF v$="s" OR v$="S" THEN GO
 TO 8540
8520 IF v$="K" OR v$="K" THEN SA
VE n$ DATA k$()
8525 VERIFY n$ DATA k$()
8530 POKE 16384,k$
8535 GO TO 1230
8540 POKE 16384,k$
8545 SAVE n$SCREEN$
8550 VERIFY n$SCREEN$
8555 POKE 16384,k$
8560 GO TO 1230
8565 REM END of save
9000 DEF PROC betolto
9005 CLS : INPUT "SCREEN- vagy t
ombtipusu ? (S/T)";l$
9010 IF l$="t" OR l$="T" THEN GO
 TO 9030
9015 IF l$="s" OR l$="S" THEN LO
AD "" SCREEN$
9020 LET k$=MEMORY$(16384 TO 1
6384+6912)
9025 GO TO 9035
9030 LOAD "" DATA k$()
9035 INPUT "Keretszin:";b: BORDE
R b
9040 INPUT "Papirszin:";p
9045 INPUT "Tintaszin:";i
9050 POKE 16384,k$: GO TO 1230
9055 END PROC
9100 DEF PROC stringek
9150 DIM m$(6,23)
9200 LET m$(1)="1.............
...DRAW"
9205 LET m$(2)="2.............
.CIRCLE"
9210 LET m$(3)="3.............
...FILL"
9215 LET m$(4)="4.........ATTRI
BUTUMOK"
9220 LET m$(5)="5.............
.TORLES"
9225 LET m$(6)="6.............
...SAVE"
9230 END PROC
```

Karakteres képernyőt másoló rutin

Ezt a kis programot azoknak ajánljuk, akik C64-es BASIC programjaik egyes képernyőit vagy ezek sorát szeretnék nyomtatón is megkapni.

A rutin három részből áll. Az első (1-15-ös sorok) tulajdonképpen magyarázat; a 10-es sor mutat példát a másoló használatára, hivására.

A második rész (500-506-os sorok) feladata az inicializálás, tehát a felhasználói programokban csak egyszer, azok elején hívjuk meg az 500-as sorban kezdődő szubrutint.

A harmadik rész (509-523-as sorok) maga a rutin, amely a képernyő másolását elvégzi a REM utasításokban megadott módon. Figyelem! A lista 517, 518-as soraiban lévő inverz záró idézőjel a lista nyomtatásához használt EPSON GX-80 nyomtató illesztőjének hibája, gépeléskor oda normál idézőjel kerüljön!

Ha valaki mindig teljes képernyőt akar másolni, az egyrészt törölje a lista 510-es sorát, másrészt az 511-es sorban KE helyére 1024-et, VE helyére pedig 2023-at írjon. Ekkor az inicializálást követő hívások is J=1, 2, vagy 3,:GOSUB509 alakra egyszerűsödnek.

FÖLDI ENDRE

```
1 REM *** HCOPY 0.1 ***
2 REM (C) EDE SOFT 1986
3 REM J=1; NO INVERZ (1.41)
4 REM J=2; "-->'      (1.66)
5 REM J=3; TOTAL      (1.66<)
6 REM KS=KEZDOSOR, VS=VEGSOR SORSZAMA
7 REM 0 <= KS <= VS <= 24
8 :
10 GOSUB500:J=3:KS=0:VS=24:GOSUB509:END
11 :
12 REM A 10-ES SOR EGY HIVASI PELDA
13 REM AZ 500-506 SOROK AZ INICIALIZALO
14 REM RESZ, TEHAT CSAK EGYSZER, A FEL-
15 REM HASZNALOI PROGRAM ELEJEN KELL A
16 REM  GOSUB500  HIVAST ELVEGEZNI!!!
101 :
500 FORC=0TO1:V$(C)=CHR$(8):RESTORE
501 FORD=0TO5:READA:A=ABS(S+A)+128
502 V$(C)=V$(C)+CHR$(A):NEXT
503 V$(C)=V$(C)+CHR$(15):S=-127:NEXT
504 FORC=0TO3:READK(C):NEXT:RETURN
505 DATA  0,  7,  0,  7,  0,  0
506 DATA 64, 32,192,224
507 :
508 :
509 CLOSE4:OPEN4,4,0,CHR$(15):CMD4
510 KE=1024+KS*40:VE=2023-(24-VS)*40
511 FORC=KETOVESTEP40:FORD=0TO39
512 P=PEEK(C+D):P1=PAND127
513 P%=P1/32:A=K(P%)+P1-P%*32
514 ONJGOTO520,516,516
515 GOTO523
516 I=PAND128:IFI=IRTHEN519
517 IR=I:I$="■":REM CTRL+9
518 IFI=0THENI$="■":REM CTRL+0
519 IFA=34THENA=39:ONJGOTO515,520,521
520 PRINTI$CHR$(A);:GOTO522
521 PRINTV$(I/128);
522 NEXT:PRINTCHR$(13);:NEXT
523 PRINT#4:CLOSE4:RETURN
```

A zűrza

Az IBM PC, XT és AT típusú számítógépek nagy sikere többek között azt is eredményezte, hogy számos cég elkészítette IBM-másolatát. Ezeket az utánzatokat — melyek szinte minden szakmai hirdetésben szerepelnek már — gyűjtőnéven klónoknak is hívják. Bizonyos nézetek szerint „ez az igazi", és hogy a hatás nagyobb legyen, a reklámok az „IBM kompatibilis" varázsigét harsogják.

A potenciális vásárlók tanácstalanok: ragaszkodjanak-e az eredeti IBM-hez, vagy bízzanak a másolatforgalmazók ajánlataiban?

Főleg kedvező áruk miatt szinte elárasztották az USA piacát a Tajvanból és Hongkongból importált gépek. Alig több, mint egy év alatt az MS—DOS-t használó utánzatok ára az IBM árak 90 százalékáról 40 százalékára csökkent. Ehhez persze azt is hozzá kell fűzni, hogy a minőség és az oly nagyon hirdetett IBM kompatibilitás egyáltalán nem garantált.

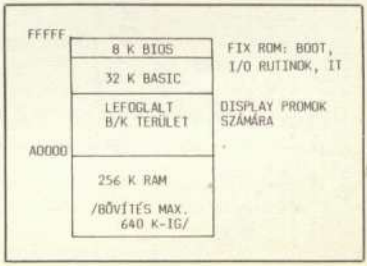
A PC kategóriájú gépek minősítése általában nem könnyű; most mégis megkíséreljük a rendelkezésünkre álló ismeretanyag alapján.

Az elsőként gyártott IBM PC, PC jr., XT, AT típusokat az IBM által felhatalmazott forgalmazók és a cég saját központjai árusították. Az árakat egységesen határozták meg, a kereskedelmi ár mindenkor a kereslet-kínálat viszonyától függ.

A vevő szempontjából a legelőnyösebb ár úgy alakul ki, ha a minimális memóriakiépítésű alapgéphez bővítéseket, első, később második floppyegységet, esetleg Winchester-lemezt, majd egy nem IBM nyomtatót és videomonitort vásárol.

Hogyan lehetséges ez? Egy nyugati kereskedelmi fogadás adja a magyarázatot. Ha ugyanis az ügyfél egyenként vásárolja meg az imént felsorolt eszközöket, és ezzel lendít a kereskedő forgalmán, akkor leggyakrabban egy merevlemezzel kedveskednek neki ajándékul. A beépített tápegység és néhány kézikönyv pedig természetes ki-

1. ábra. A PC—DOS tárfelosztása



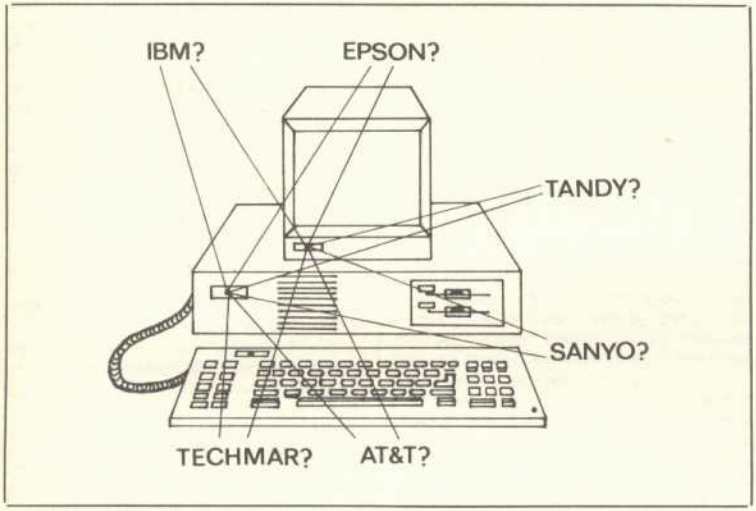
rtól a zsibongásig

-utánzatok a piacon

egészítők. Az IBM mintaboltban egyébként az érdeklődő B/K kártyákhoz, videoadapterekhez, PC—DOS verziókhoz és hatalmas mennyiségű felhasználói szoftverhez is hozzájuthat.

Említettük már, hogy ha a vevő nem elég óvatos, könnyen becsaphatják egy ázsiai másolattal, viszont az IBM által nem feljogosított kereskedőknél is bárki hozzájuthat valódi IBM áruhoz. A vásárlót ez még jobban összezavarhatja, hiszen a tapasztalatok szerint itt sem kell rosszabbra számítania, mint az engedélyezett üzletekben.

De honnan kerül az illegális eladókhoz eredeti IBM eszköz? Természetesen a felhatalmazott eladók polcairól. Az indok is



2. ábra. Zavar a zűr?

kézenfekvő: a kereskedő részesedést kap az eladott darabszám és a forgalmazás sebessége után.

A piaci szemlét folytassuk a legjobb minősítésű MS—DOS rendszerű klónoknál.

Nem sokkal az IBM PC megjelenése után a Compaq Company megjelentette az első IBM kompatíbilis PC-t. Ez még nem tekinthető utánzatnak, ugyanis hordozható kivitelben készült, és minden tekintetben felvehette a versenyt az IBM géppel. A Compaq képviselői sokáig hangoztatták, hogy az ezután készült hordozható IBM gépek tulajdonképpen Compaq kompatíbilisek. A Compaq nagy sikert ért el asztali számítógépeivel, és azóta is megtartotta pozícióját. Később már amerikai PC-klón gyártók tűntek fel, többek között a Sequia, Columbia, Eagle, Corona, Tandy és az AT&T. De a kisebb vállalkozások nem maradhattak sokáig felszínen az IBM-mel szembeni legalizálási és technikai problémák miatt. A Sequia, a Columbia és az Eagle is így szorult ki a piacról. A Corona több különböző vállalatnak kínálta termékeit, a Tandy saját kereskedelmi hálózattal próbálkozott, az AT&T pedig igen jelentős piaci keresletet kötött le magának saját tervezésű mikrogépeivel.

Napjainkban az igazán nagy gyártók — például: AT&T, ITT, Sperry, Hewlett—Packard, Heath/Zenith, Fujitsu, EPSON, Tandy, Canon — szinte kivétel nélkül IBM PC kompatíbilis gépeket bocsátanak ki. Ezek között vannak olyanok is, melyekben nagyobb a memória, gyorsabb a rendszer órajele, mások viszont még hardver szempontból is gyengébbek, mint az IBM. Kiemelkedő szoftverekkel pedig csak a Hewlett—Packard és a Tandy büszkélkedhet.

Ha már a legjobbakról esett szó, felvetődik egy kérdés: milyen szigorúsággal értelmezik a kompatíbilis kifejezést, és milyen fokú kompatibilitás érhető el? Manapság a kompatibilitás nem vehető minden gyártó szájából garanciának. Habár az IBM programok többsége lefuthat egy klónon is, a fennmaradó szoftveranyagra nincs biztosíték. Egyértelműen kijelenthető — még ha a kompatíbilis szónak csökkent is a jelentősége —, hogy százszázalékos IBM-másolat nincs! Az IBM ugyanis egyik MS—DOS

verzióját, a PC—DOS-t egy ROM-ban helyezte el, és ezt az operációs rendszert hivatalos szerzői joggal védi (lásd az *1. ábrát*). Ebben a ROM-ban található egy BASIC verzió és sok egyéb járulékos rendszerrutin. A ROM-nak ezt a részét ROM BIOS-nak (Basic Input/Output System = Alap I/O Rendszer) hívják.

Ha egy IBM PC/XT bekapcsolásakor nem teszünk a meghajtóba egy DOS lemezt, akkor a ROM-ban levő BASIC áttöltődik a RAM-ba, majd a vezérlés is ide kerül. A PC—DOS esetében a beégetett BASIC vagy BASIC-A kiegészül a floppyn lévő rutinokkal. Ha a BASIC ROM hiányzik, hibajelzést kapunk.

Éppen a védettség miatt más MS—DOS gépekben nincs BIOS—ROM. Az ilyen gépekkel rendelkező felhasználókat a MICROSOFT DOS verziói segítik. A BASIC-et — mely természetesen azonos az IBM fordítójával — az említett operációs rendszer alatt lemezről kell betölteni. Ezt hívják egyébként GW BASIC-nek.

Az a szoftver, mely az IBM ROM-jában lévő rutinok közül akár egyet is meghív, már nem fut le egy klónon.

Néhány gyártó készített 16 bites Intel processzora épülő MS—DOS gépeket, de ezek nem voltak IBM PC kompatíbilisek. Ebbe a csoportba tartoztak a TI Professional, a DEC Rainbow, a Victor 9000, az Epson QX—16, a Tandy Model 2000, az Apricot és a Sanyo—555 típusú számítógépek. Szoftverjeik hasonlítottak a már megszokott PC programokhoz, de saját tervezéssel kellett hogy készüljenek.

A Sanyo—555 a PC kategóriában nagy sikereknek örvendett. A cég 1985-ben egy új videoadapter-egységgel rukkolt elő, s ezt már az 550-es szériába — mely közelebb áll az IBM PC-hez, mint elődje — be is építették. A Tandy 2000 pedig, amely igazi 16 bites központi egységre, a 80186-ra épülő és ezáltal a 8 MHz-es órajelet használó gép, az IBM PC vetélytársává vált: ez ugyanis a 4,7 MHz-es órával többek között csak 8 bites külső adatbuszt kezelt.

A T. Model 2000 memóriája ráadásul 256 k-ról 768 k-ra bővíthető, szemben a megszokott 640 k-s határral. Ebből következik a jobb, 640×400-as grafikai felbontás is, ami függőlegesen kétszerese az IBM PC felbontásának. A Tandy sem volt IBM kompatibilis; ez is hozzájárult ahhoz, hogy csak jóval a hardver után készültek el a szoftveranyagok. Az IBM tehát itt is a programok óriási mennyiségével nyerte meg a csatát. Később Tandy is beadta a derekát, és bár a legtökéletesebb termékeket kínálja, azok mégis másolatokká alakultak át.

Az Angliában készülő Apricot gépek listáján az olcsó FI modelltől a 995 S-os XEN-ig több más, komoly gép is található. A PC-khez viszonyított számos eltérés közül a leglényegesebb, hogy az Apricot MS—DOS gépei 3,5" méretű floppykat kezelnek.

A nem IBM kompatibilis gépek története tehát eléggé szomorú. Még a második legnagyobb cég, a DEC is csak szerény piaci sikereket könyvelhetett el; az EPSON QX—16 és a TI Professional kudarca kétségtelen. Számítástechnikai léptékkel mérve azóta sok idő telt el, és ma a Tandy, az EPSON és a Sanyo a legjobb PC/XT/AT klónokat bocsátja piacra. Ha valaki mégis a már említett, nem kompatibilis gépekből választ, akkor az amerikai kritikák a TI Pro-t és a Tandy 2000-et javasolják. Ezekhez időközben jelentős mennyiségű szoftver született.

A vásárlók általában nincsenek tisztában azzal, hogy milyen fizikai szolgáltatásokat várjanak egy PC-től. Vegyük hát sorra ezeket.

Elsőnek nézzük a memória kiépítettségét, a RAM-kapacitást! „A felhasználónak mindig és mindig nagyobb tárra lenne szüksége, mint amekkorával rendelkezik" — mondják sokan. 30 évvel ezelőtt csak 8 k-s szabad kapacitással lehetett számolni, és a felhasználó örült a 16 k-s, később a 32 k-s, majd a 64 k-s bővítőkártyáknak. Napjainkban a legkisebb PC alapmemória 256 kbájt, 128 kbites chipekből összeállítva. Az MS—DOS 3.1 akár 640 k RAM-ot is kezel. A forgalmazók már a kezdéshez is 640 k-t javasolnak, árengedményeket ígérve, ha a felhasználó 960 k-ra bővít. Ma már az USA-ban a 256 kbites chip ára alacsonyabb, mint a 128 kbitesé!

A következő szempont a tápegységek terhelhetősége. Az eredeti IBM PC-kben például 70 W teljesítményt garantáltak. Amennyiben beépíthető merev- vagy hajlékonylemez-egységgel bővül a rendszer, a tápegység nem biztosít elegendő energiát. A legálisan forgalomba hozható egységeket az U. S. Underwriters Labs nevű intézmény hagyja jóvá. A szakértők a kedvezőbb árak ellenére óvnak az „illegális" tápegységektől.

Ejtsünk néhány szót a floppymeghajtókról is. Az IBM PC-kben kezdetben teljes magasságú meghajtó volt. A félmagasságú egységek térhódítása nyomán két floppyval lett gazdagabb a rendszer. Az egyoldalas meghajtók átalakultak kétoldalasakká; a másik oldal eléréséhez nem kell megfordítani a lemezt. A mai operációs rendszerek mellett pedig szinte bűn egyetlen floppyval dolgozni, már csak azért is, mert ez könnyen kiváltható egy nagy kapacitású Winchester-lemezzel.

A fixlemezek anyagfelhasználás és önköltség szempontjából azonos szinten mozognak, áruk a piacon mégis óriási szórást mutat. A már említett ajándék lemezek esetében viszont igenis érdemes megnézni az „ajándék ló fogát"! 1986-ra a 10 vagy 20 Mbájtos egységek jöttek divatba, és áruk is mérséklődött. Ajánlott gyártók a BASF és a Greshams Law.

Nagyon fontos a PC-k kapcsolata a külvilággal. Feltétlenül szükség van egy párhuzamos printer portra és egy soros RS232 B/K portra. Ha egér vagy botkormány is van, akkor két darab RS232 sem árt. Néhány klóngyártó az alapkártyán helyezte el a B/K portokat, de a legújabbakhoz — követve az IBM-et — már külön csatolókártyát lehet kapcsolni. Ez természetesen főleg üzleti fogás, bár a slotok csatlakozói többfunkciósak is lehetnek, hiszen a perifériavezérlő jelek mellett gyakran a központiegység- és rendszervezérlő jeleket is kivezetik.

Ami a hardvertermékeket illeti; meg kell állapítanunk, hogy azok szabadalmaztatott védelmével jóval kevésbé törődnek, mint a szoftver szerzői jogokkal. A másolat-NYÁK-ok minősége a másolatgépekénél is sokkal rosszabb; a Quadram, STB vagy Technar négyrétegű NYÁK-ra 30—90 napos jótállás van, szemben a 8—14 hónapos IBM-garanciával.

Utoljára hagytuk a videomonitorok területét. Az eredeti IBM PC-nek fekete-fehér, nagy szövegfelbontású, de nem teljes grafikus monitora van. Az U. S. Company Hercules nevű gépéhez már teljes grafikus adaptert adtak, később pedig alapvető igény lett a színes monitor.

A vásárlók gyakran meghökkennek, hallván, hogy egy monitor nem alkalmas akármilyen PC-hez. Pedig így van! Felesleges a jó felbontású színes monitor, ha a számítógép hardvere (video RAM+colour áramkörök) nem képes támogatni, és viszont: rossz monitorral nem aknázhatjuk ki a gép lehetőségeit. A jelenlegi „közepes" RGB, illetve NTSC monitorok: GRAPHICS HX12, AMDEK COLOR 600, 700, 710, TAXAN 640, IBM. A legjobbaknak a Zenith és a Toshiba termékeit tartják. Ár szempontjából az USA-ban az a legkedvezőbb, ha egy klón számítógéphez márkás monitort vásárol a felhasználó.

1986 végére akadtak olyan hirdetések is, melyekben PC másolat kiteket kínáltak. Ha ezekből állít össze valaki egy konfigurációt, még több pénzt takaríthat meg. Az összeszerelés viszont tapasztalatot és nagy gyakorlatot igényel.

Tehát sokféle lehetőség közül választhatunk, beszerzés előtt azonban érdemes tájékozódni. Reméljük, hogy írásunkkal sikerül a külföldi eligazodáshoz segítséget, a hazai helyzet megítéléséhez pedig bizonyos szemléletet nyújtani.

HARGITAI PÁL

Az érdeklődés megoszlása

Általános tapasztalat, hogy a konferenciákon elhangzottak nem minden résztvevőt érdekelnek egyformán. A többség csupán a számára érdekes egy-két előadást szeretné meghallgatni, és a konferencia írásos anyagát is gyakran azért nem vásárolják meg, mert legnagyobb részére nem kíváncsiak. A távkonferenciázás révén a hallgatóság az előadások anyagában kedvére válogathat — azaz mentesülhet a számára érdektelen előadások meghallgatásától —, ami azért is előnyös, mert a felszabaduló időt is munkájával töltheti.

Érdekes ellentmondás, hogy a kormányok ugyan támogatják a nemzetközi turizmus különböző formáit, így a konferenciaturizmust is, de az információtechnológiai fejlesztések, köztük a távkonferenciázás támogatásával egyúttal fékezik is azt. Ha mindenképpen prioritást kell megállapítani, akkor az információtechnológiának a konferencia-turizmussal szemben elsőbbséget kell élveznie.

A nemzetközi együttműködés jelentősége

Az ipar felismerte, hogy nemzetközi együttműködéssel hatásosabban léphet fel harmadik piacokon. Ezért a fejlett nyugat-európai tőkésországok (Anglia, NSZK, Franciaország, Olaszország) 1983-ban létrehozták az Európai Videokonferencia-hálózatot, Anglia és az USA pedig a két ország közötti kétirányú videokapcsolatot megteremtő Satellit Business Systemet. Ezek — a gazdaságossági szempontokon túl — jól szolgálják az egységesítési (szabványosítási) törekvéseket is.

A távkonferencia- és a személyiszámítógép-ipar összefonódása

A két szakterület összefonódása mind a gyártók, mind a felhasználók körében megfigyelhető (gyártmányválaszték, installációk). Az összefonódás okai az alábbiakban keresendők:

— új rendszerek kiépítésének lehetősége a már meglevő eszközbázison;

— PC-tulajdonosok igyekezete, hogy saját gépeikről a szolgáltatások minél szélesebb köréhez jussanak;

— az egyre bővülő szolgáltatások.

A kereskedelem és a potenciális érdeklődők jelenleg várakozó álláspontot foglalnak el, és figyelik a különböző alkalmazá-

ávkonferencia II.

sok felhasználói visszhangját, illetve fogadtatását.

A szakemberek óvnak a távkonferenciázás „túlértékelésétől" és az esetleg ebből fakadó későbbi csalódásoktól, mivel úgy vélik, hogy a felhasználók részéről tapasztalható erőteljes igény nem annyira magának a távkonferenciázásnak szól, mint inkább saját feladataik hatékonyabb ellátása iránti érdekükből fakad.

A rendszerek „sokoldalúságát" néhány példával érzékeltetjük.

A fokozatosan változó képet továbbító (freeze frame) Infórum rendszerben egy időben többféle tevékenység is végezhető: kép- és hangtovábbítás, adatfeldolgozás, csíkkódos adatbevitel, táblázatfeldolgozás stb. A hálózat állomásai IBM PC XT gépek.

A Travelers rendszert leginkább távértekezletekre, illetve oktatási célokra használják; jelenleg még csak a hatfordi, a narcrossi és az orlandói kirendeltségeket kapcsolták össze, de további bővítéseket is terveznek.

A mozgóképes távkonferencia-rendszerek terjedését a „húzó" tényezők ellenére ma még fékezik a meglehetősen magas beruházási költségek (közvetítőtermi és vonalbérleti költségek), továbbá az a tény, hogy az esetek többségében valójában nincs is szükség a mozgókép-közvetítésre. Nem közömbös, hogy a távkonferencia-rendszerek megvalósítási, illetve bérletezési (vonal) költségeit összehasonlítva, a freeze frame-es megoldással kapcsolatos költségek — főleg az egyszerűbb műszaki megoldások miatt — a mozgóképesnek alig 10%-át teszik ki.

Az IBM rendszere

Az IBM videokonferencia-hálózata IBM PC XT gépeket kapcsol össze freeze frame típusú rendszerré, mely a konferenciateremből fekete-fehér képet közvetít. A rendszerbe való belépés tárcsázással történik. Az átviteli közeg a kapcsolt hálózat; a dokumentációkról 3-5 másodpercenként, a teremről 8-10 másodpercenként továbbítanak egy-egy képet. A továbbított képet és hangot átmenetileg tárolják, így ezek 5 másodpercen belül visszahívhatók.

Az IBM rendszertől — a nagyképes és PC-s gyakorlatnak megfelelően — „de facto" szabvánnyá válását várják a szakemberek.

Audiokonferencia-rendszerek

Az összes távkonferencia-rendszerek kb. 90 százalékát a csak hangkapcsolatot teremtő rendszerek teszik ki: a kapcsolatteremtés eszköze a telefonkészülék, az elhangzottakat mindenki hallja, és közbe is szólhat. Az ilyen rendszer létesítésének, üzemeltetésének költségei a legalacsonyabbak, és leginkább arra valók, hogy távoli munkahelyeken dolgozó munkatársakat gyűjtsenek össze értekezletekre.

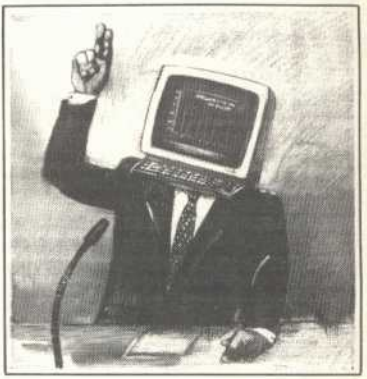
A hangkapcsolatú munkaértekezletek időtartamát vizsgálva kiderült, hogy 90 százalékuk maximum 30 percig, 67 százalékuk 50 percig, 8 százalékuk pedig 2 óránál hosszabb ideig tartott. Általános tapasztalat, hogy a távértekezletek és távkonferenciák időtartama jelentősen rövidebb, mint a hagyományos értekezleteké és konferenciáké.

Működő videokonferencia-rendszerek

A jelenleg működtetett távkonferencia-rendszerek száma — akár műszaki megoldásaikat, akár szolgáltatásaikat tekintve — igen nagy. Az egyes rendszereknek csupán a felsorolása is kötetnyi; az alkalmazásbeli lehetőségek bemutatására utaljon néhány szó a sikeresen működő rendszerekről.

A Bears műholdas összeköttetéssel (American Satellit) működő videokonferencia-rendszere „panoráma" formátumú (széles) képet közvetít a rendezvényekről, így küszöbölve ki a konferenciaterembeli események követéséhez szükséges meglehetősen drága, többkamerás, képváltásos megoldást. A rendszer a teremképen kívül az előadás nyomtatott anyagáról is jó minőségű állóképet közvetít, továbbá biztosítja a kétirányú, párbeszédes jellegű hangkapcsolatot.

A Xerox vállalati munkaértekezleteinek lebonyolítására, a Penney New York-i részlegeinek összekapcsolásával megtartott termékismertetőkre, a McDonald Douglas pedig fiókvállalatainak naprakész információkkal való folyamatos ellátására rendszeresítette a távkonferencia-szolgáltatást. A Hilton szállodahálózat több városra kiterjedő — 35 szállodáját összekapcsoló — hálózatot működtet, melynek segítségével egyszerre 6 helyszínen folyó különböző rendezvényekről ad álló- és mozgóképet, illetve hangot. A Holiday Inn szállodakonszern hálózatához az USA-n belül és kívül 349 állomás tartozik, és a rendszerrel 1984-ben 50 konferenciát tartottak 175 ezer résztvevő számára. A Federal Express a rendezvényeiről a távkonferencia segítségével a Londont, Brüsszelt, San Juant, Honolulut, Torontót átfogó körzetben (240 állomás) kb. 25 ezer személy számára közvetített.



Napjaink egyik legkorszerűbb mozgókép-továbbító rendszere a Bell Laboratórium által kifejlesztett VIVID rendszer, amely igen jó minőségű színes mozgóképet (teremkép), dokumentációkról nagy felbontású állóképet és HiFi minőségű sztereo hangot továbbít, elsősorban az egy városon belüli intézmények számára. A rendszer egyik előnye, hogy minimális üzemeltetői beavatkozással működtethető, és képes mind a két helyszín közötti „pont-pont" jellegű, mind az egy helyszínről sok helyszínre szóró „broadcast" típusú közvetítésre, fénykábelen vagy a hagyományos távközlési hálózaton keresztül. A VIVID rendszer ára a szabványosításnak és a magas színvonalú technológia alkalmazásának köszönhetően meglepően alacsony; a konferenciatermi berendezések beruházási költsége nem éri el egy másik rendszer egyetlen videokódolójának az árát sem.

Az iparág pénzügyi „fejlődése"

A távkonferencia-ipar forgalma 1983-ban 370 millió USA dollár volt; ebből az audiokonferencia-rendszerek részaránya 300 millió, az egyirányú átvitelt biztosító videokonferencia-rendszerekéé 19 millió, a szakaszosan változó képet továbbító rendszereké 1 millió, a képtelefon-szolgáltatásé 50 millió.

Az iparág 1992-re jósolt összbevétele 3,4 milliárd dollár. Ebből a képtelefon-szolgáltatás részaránya 1,6 milliárd, az audiokonferencia-szolgáltatásé 1,1 milliárd, az egyirányú videokonferenciáé 730 millió, a sza-

kaszosan változó képet továbbító rendszerré 16 millió, az összes többi pedig 3 millió dollár. Csak a különböző videokonferencia-berendezések forgalma az 1983-as 8 millió dollárról 1992-re várhatóan 229 millió dollárra nő.

A leggyakrabban alkalmazott videokonferencia-rendszerek jelenlegi és tervezett installációinak alakulását a *táblázat* szemlélteti.

Szakkiadványok

A Visconsin egyetem a nagy érdeklődésre való tekintettel jelenteti meg a Telecon Newslettert, amely profi módon foglalkozik a különböző távkonferencia-rendszerekkel s bennük az új rendszertechnikai megoldásokkal, és időnként bemutat egyket sikeres alkalmazást. Az újság ismerteti a különböző termékek mindenkori piaci helyzetét is.

Egy másik kiadvány, a távkonferencia-felhasználók „telefonkönyve" tartalmazza mindazon felhasználók (több mint 300 intézmény) nevét, címét és telefonszámát, amelyek valamilyen formában használják a távkonferencia-rendszerek egyikét vagy másikát, illetve igénybe veszik e rendszerek szolgáltatásait. A telefonkönyv sorolja a különböző szolgáltatásokat kínálók, valamint a készülékgyártók címjegyzékét is.

Rendszerkiválasztás

Valamely funkciónak, feladatnak a megoldása, illetőleg a megoldásnak a módszere lehet egyfajta távkonferencia. A döntés azonban alapos körültekintést igényel: elemezni kell magát az elvégzendő feladatot, az intézménynek a távkonferencia-rendszerrel kapcsolatos pillanatnyi és a jövőbeni elvárásait, valamint a fejlesztés által elérhető nyereséget vagy megtakarítást, és végül a beruházás költségeit (a teljesítmény/költség hányados optimalizálása).

Vizsgálni kell az intézmény profilját, szervezeti felépítését, kommunikációs ellátottságát, magát a közvetítendő rendezvény jellegét (milyen típusú közvetítés a célszerű, szükséges-e az állandó operátori felügyelet stb.), a távkonferenciával „összekapcsolandó" részlegek egymástól való földrajzi távolságát, a hallgatóság várható számát, az intézménynél már meglévő gépparkot és a munkába bevonható szakszemélyzet képzettségi színvonalát. Kulcskérdés, hogy a feladat milyen típusú kapcsolatteremtést igényel (álló/mozgó kép, hang továbbítása, egy-, illetve kétirányú átvitel stb.) A szóba jöhető rendszerek közül a szükségesnél többet nyújtó túl drága, amelyik pedig annál kevesebbet tud, azt nem érdemes megvenni.

A rendszer kiválasztásakor nem lehet megkerülni a más hálózatokkal való összekapcsolhatóság kérdését, a különböző szabványokhoz való illeszkedést (PAL, SECAM, NTSC). Fontos szempontok: a szervizelhetőség, a közvetített kép minőségét befolyásoló tényezők, az egyszerű kezelhetőség, a különböző közvetítési környezetekhez, valamint a felhasználói igényekhez (szín- és képminőség) való rugalmas alkalmazkodóképesség (mikrofonok, hangszórók, kamerák, monitorok választéka, világítási igény, elhelyezés stb.).

A jól működő távkonferencia-rendszer létesítéséhez szervesen kapcsolódik a konferenciaterem kialakítása; ennek költségei meglehetősen magasak, ezért az intézmények inkább a nélkülözhető helyiségeiket igyeknek konferenciateremmé átalakítani. Ez azonban nem mindig sikerül optimálisan. Sokat segítene egy jól átgondolt ajánlás, ami lényegesen csökkentené a teremkialakítás ma még tetemes költségeit. Feltétlenül javítani kellene a már meglevő konferenciatermek kihasználtságán — elsősorban szervezett propagandával. A potenciális ügyfelek megnyerésére ugyanis a leghatásosabb reklám az új megoldás ismertetése. Be kell mutatni annak előnyeit és hátrányait, hogy mire használható, melyek a buktatók, mire érdemes vigyázni, és mindezt lehetőleg egy működő rendszeren, „élesben".

Perspektívák

A vállalati életben tapasztalható decentralizációs törekvések erősödésével egyre inkább nő a távkonferencia-rendszerek szerepe, elsősorban az értekezletekre való utazgatásokkal eltöltött idő csökkentésében. A másik igen fontos tényező a vállalati döntéshozatali mechanizmus biztonságának növelése, megalapozottabbá tétele — mindez a naprakész informálás javításával —, és ezáltal a piacgazdálkodás rugalmasságának fokozása. Egy lehetséges további cél (nyereségforrás) a vállalati adminisztráció csökkentése, és egyúttal a központi utasítások gyorsabb eljuttatása a kihelyezett részlegekhez, például elektronikus levelezési rendszer segítségével.

* * *

A távkonferenciázás számos vetületét nem említettük cikkünkben. Gondoljunk csak a biztonsági szempontokra, vagy — és ezzel szemben — az elidegenedésre. De mindenről nem szólhattunk. Reméljük azonban, hogy olvasóink érdeklődését felkeltettük a jövő e — talán hamarosan honosítható — lehetősége iránt.

CSEH KÁLMÁN

ADOK—VESZEK —CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közleteknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,-Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

C64-hez EGÉR, ROM-diszk, TAPE-COPY (magnó-felvétel duplikáló), Spectrumhoz Joystick-interfésszel, valamint printer- és floppyillesztő eladó. Telefon: 663-528, 460-832.

Commodore VC20-as programok eladása: Assembler, Disassembler, Finomgrafika stb. Kérjen díjtalan tájékoztatót! Levélcím: Juhász György, Salgótarján, Pf.:157. 3100.

ZX-Spectrum és ZX81 BASIC és Assemblerrel foglalkozó könyvek eladók, vagy a Mikroszámítógép Magazin számaira cserélem. Plenter Róbert, Nyíregyháza, Toldi u. 42. 4400.

ZX-Spectrum (48 k) + INTERFACE 1 + 2 db MICRODRIVE + 6 db Cartridge + ZX - PRINTER 4 tekercs papirral, magyar nyelvű irodalmakkal eladó. Ár: 35 000,- Ft. Telefon: 897-358 du. 4 óra után.

VESZEK

VC20-as számítógépre mindenféle hardver és szoftver érdekel. Árajánlatokat és rövid programleírást kérek. Különösen érdekel 64 It-s és 35 It-s bővítő gépbe való beépítése vagy külön CARTRIDGE formában. Gádor József, Jászfényszaru, Hunyadi J. út 19. 5126.

CSERÉLEK

C64-es programokat cserélek kazettán. Varga Károly, Kapuvár,Lenin u. 11. 9330.

C64-es programokat cserélek kazettán. Címem: Gerots Lajosné, Vasvár, Táncsics u. 9. 9800.

C64-es rajz-, zene-, szöveg-, játékszerkesztő, repülés szimulátor programokat, komoly játékprogramokat keresek cserével lemezen. Pekárovics Zoltán, Debrecen, Holló J. u. 4. II. 14. 4024.

Commodore Plus/4-es játékprogramokat cserélek. Stefancsik Tamás, Szentendre, Dózsa György u. 19. 2000.

Videoton TV-Computer tulajdonosokat keresek program- és tapasztalatcsere céljából. Assemblert, disassemblert keresek. Solti Csaba, Szeged, Kossuth L.u. 72/B. 6724.

ZX-Spectrum (48 k) játékprogramokat adok-veszek-cserélek. A válaszokat listával kérem: Duró Zoltán, Berettyóújfalu, Nádasdi u. 2. 4100.

ZX-Spectrumhoz (48 k) programokat cserélek. Válaszokat a programok listájával együtt kérem: Nagy Szabolcs, Pacsa, Csányi László u. 55. 8761.

ZX-Spectrumhoz programokat cserélek. Válaszokat a programok listájával együtt erre a címre kérek: Kató János, Mezőhegyes, Cukorgyári ltp. 2. 5820.

μINFORM

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

A folyóirat neve	Kódja
64'er Magazin	64er
Chip Magazin	chip
Commodore Horizons	coho
Commodore Microcomputers	comi
Compute!	cute
Computer Persönlich	pers
Happy Computer	happ
hc - Mein Home-Computer	hc
mc - Zeitschrift	mc
Run /USA/	run
Sinclair User	sinc
Your Sinclair	ysin

A tartalomleíró szóláncokat permutáltuk, a szólánctvariánsokat pedig alfabetikusan rendeztük.

A tartalomleírás egy szóláncból áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkeréséséhez a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szóláncában elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szűkebben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karaktersorozatát nyíl vezeti be, melyet a / jelig a folyóiratok azonosítója, a két / jel között az évszám, folyóiratszám és kötőjellel a kezdő oldalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

A folyóiratok a SZÁMALK szakkönyvtárában (Budapest XI., Szakasits Á. út 68. Nyitva: 8-tól fél 5-ig. Tel.: 853-111/251) is föllelhetők. A kiválasztott anyagról másolat rendelhető az alábbi formában:

SZÁMALK Szakkönyvtára
Budapest, 112. Pf.: 146. 1502

Megrendelem a Mikroszámítógép Magazin 1987/ sz. alapján a következő folyóirat-oldal-másolatokat:

Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____

A megrendeléshez csatolom az oldalankénti 8,- Ft-os szolgáltatási díj befizetését igazoló csekkszelvényt.
Dátum, név, pontos cím.

PROGRAMLISTA
adatrendezes▮mbasic programozas▮shell-metzner algoritmus alkalmazasa
->pers/86.07-74/3
PROGRAMLISTA
apple ii▮<shapetable-editor>
->mc/86.03-70/2
PROGRAMLISTA
apple ii▮barkacsolas▮commodore 64▮programcsere lehetoseg
->chip/86.04-161/2
PROGRAMLISTA
apple ii▮illesztoprogram
->mc/86.04-78/2
PROGRAMLISTA
apple ii▮merestechnika▮<interpolacios polynom> ->pers/86.09-71/2
PROGRAMLISTA
apple ii▮nyomtato(fx-80)▮pascal programozas▮turbo-pascal vezerlo modul
->pers/86.08-69/5
PROGRAMLISTA
apple ii▮nyomtato(mx-80)▮pascal programozas▮hires grafika assembler rutin nelkul ->mc/86.03-64/2
PROGRAMLISTA
apple iic▮commodore 64▮ugroutasitas-cim automatikus modositasa atszamozaskor ->mc/86.03-72/3
PROGRAMLISTA
apple iie/+▮merestechnika▮polinom generalas meresi pontokhoz
->pers/86.10-43/3
PROGRAMLISTA
atari 600/800xl/130 xe▮operacios rendszer rutinok▮hozzaferes basicbol
->chip/86.04-170/2
PROGRAMLISTA
atari xl/st▮120 karakteres futoszoveg generalasa ->hc/86.04-70/1
PROGRAMLISTA
atari xl/st▮cursor-delete-insert aktivalas az input-hoz ->hc/86.04-68/1
PROGRAMLISTA
atari xl/st▮digitalis ora generalasa a 24 alapsor fole ->hc/86.04-70/2
PROGRAMLISTA
atari xl/st▮informaciosor a 24 alapsor felett▮<info line>
->hc/86.04-69/1
PROGRAMLISTA
atari xl/st▮jatekkeszito segedprogram ->hc/86.04-72/8
PROGRAMLISTA
commodore 128▮<autochange>▮? modusvaltas boot-fazisban
->64er/86.04-84/2
PROGRAMLISTA
commodore 128▮programozasi fogasok▮programvedo poke ->64er/86.04-81/2

PROGRAMLISTA
commodore 16:64▮szinbeallitasok▮c-16 basic-utasitas szimulacioja
->hc/86.04-38/3
PROGRAMLISTA
commodore 64▮<charmove>▮sprite vezerles ->hc/86.04-30/3
PROGRAMLISTA
commodore 64▮<runscript 64>▮szovegszerkeszto rendszer gepi kodban
->run/86.03-40/11
PROGRAMLISTA
commodore 64▮directory behivas programfeluliras nelkul
->coho/86.04-54/1
PROGRAMLISTA
commodore 64▮grafika(hires)▮szines kepek fekete-feher-szurke nyomtatasa
->64er/86.04-67/4
PROGRAMLISTA
commodore 64▮kiegeszites az elozo szam <dos-gen> anyagahoz
->64er/86.04-10/1
PROGRAMLISTA
commodore 64▮modulok menuvezerlesu osszekapcsolasa▮<hypra-basic>
->64er/86.04-58/6
PROGRAMLISTA
commodore 64▮muveletgyorsitas▮664+104 blokk lemezreiras
->happ/86.04-68/2
PROGRAMLISTA
commodore 64▮nyomtato modul a <database>-hez ->64er/86.04-63/2
PROGRAMLISTA
commodore 64▮oktatas▮<quizmaster>▮Kerdeshalmaz generalas
->64er/86.04-53/5
PROGRAMLISTA
commodore 64▮programfutas idomerese
->64er/86.04-85/5
PROGRAMLISTA
commodore 64▮programozasi fogasok haladoknak ->64er/86.04-78/3
PROGRAMLISTA
commodore 64▮szambillentyuk athelyezese kenyelmes data-bevitelhez
->run/86.03-50/4
PROGRAMLISTA
commodore pc-10▮input szubrutin
->pers/86.09-51/2
PROGRAMLISTA
cp/m▮help-funkcio beepitese
->pers/86.07-85/1
PROGRAMLISTA
cp/m▮lemezkapacitas es nyomtatostatus lekerdezes ->pers/86.07-72/1
PROGRAMLISTA
cp/m▮pascal programozas▮(type) program turbo-pascal valtozata
->pers/86.10-49/3

PROGRAMLISTA
dbase programozas▮listaformatalas programtomorites utan
->pers/86.10-51/3
PROGRAMLISTA
fizika▮merestechnika▮oktatas▮szemleltato kiserletek ->happ/86.04-39/6
PROGRAMLISTA
levelezes▮pascal programozas▮allando szovegmodulok hasznalata
->mc/86.04-84/4
PROGRAMLISTA
oktatas▮pascal programozas▮atteres basicrol▮a fa-struktura elemzese
->mc/86.03-66/3
PROGRAMLISTA
pascal programozas▮peldaprogram
->hc/86.04-62/4
PROGRAMLISTA
pascal programozas▮turbo-pascal kepernyokezelo eljarasok
->pers/86.07-70/1
PROGRAMLISTA
run▮beviteli utmutato es programok
->run/86.03-97/3
PROGRAMLISTA
sinclair spectrum▮jatekprogram▮<code busters>▮<mastermind> valtozat
->ysin/86.04-46/2
PROGRAMLISTA
sinclair spectrum▮keretszin beallitas ->hc/86.04-80/1
PROGRAMLISTA
sinclair spectrum▮program-megorzes reset hasznalatkor ->hc/86.04-81/2
PROGRAMLISTA
sinclair spectrum▮tizedesszam formatalas ->hc/86.04-79/2
PROGRAMLISTA
sinclair spectrum▮tv-kepernyo vedelem▮kikapcsolo rutin
->ysin/86.04-73/2
PROGRAMLISTA
sinclair zx-81▮grafika(hires)
->chip/86.04-173/3
PROGRAMLISTA
sinclair▮jatekprogram▮<cherry run>
->ysin/86.04-50/4
PROGRAMLISTA
sinclair▮karaktertervezes/modositas
->ysin/86.04-44/2
PROGRAMLISTA
terminal uzemmod▮atari st▮<contakt st> ->happ/86.04-84/2
PROGRAMLISTA
vezerles▮barkacsolas▮bitminta-kiadas led-diodakra ->happ/86.04-28/6
PROGRAMLISTA
zene▮commodore 64▮<simple sound> alprogram keszlet ->happ/86.04-70/3

Kétrészes cikksorozatunk első részében arról volt szó, hogy MS—DOS alatt futó gépek bizonyos típusú fájljai hogyan alakíthatók át más operációs rendszerű gépek által értelmezhetőkké. E második rész a fordított folyamatot mutatja be. Azoknak a géptípusoknak az esetében, amelyek a BASIC programokat ASCII formában (szövegfájlként) is képesek tárolni, majd ismét programként felismerni, ez a módszer programok átvitelére is alkalmas.

Bár a jelen számunkban közölt RAINBOW" cikket a TRS—80 Color Computer felhasználóinak írták, csak látszólag szól kizárólag a CoCo-soknak. A legtöbb mikroszámítógépben a Western Digital-cég valamelyik áramköre a lemezkezelő. Az IBM PC/XT és kompatibilis gépekben nem. Eme eltérés miatt a nem MS—DOS-t használó gépeken írt programok átvitele MS—DOS gépekre sokaknak nem sikerült. Ezek számára ad segítséget ez az írás. A cikknek, valamint az MS—DOS lemezre író programnak az áttanulmányozása hasznos az átírással foglalkozni nem kívánó olvasó számára is, ha kíváncsi a lemezkezelés egy gyakori megoldására.

CoCo szövegfájlok konvertálása MS—DOS lemezre

Reprinted from the Rainbow, The Color Computer Monthly Magazine, Copyright Falsoft, Inc., 1984, P. O. Box 385 Prospect, KY 40059

Az előző számban azt tárgyaltuk, hogyan tehetünk olvashatóvá a CoCo számára egy egyoldalas MS—DOS lemezt. Most azt ismertetjük, hogyan formálható a CoCo-n MS—DOS lemez, és hogyan vihető fel a szövegfájl a CoCo-val erre az MS—DOS lemezre.

Legalább 2 db 40 sávos lemezmeghajtóra van szükség. Egy lemezmeghajtóval ugyanis a konverzió elfogadhatatlanul lassú lenne. Lényegtelen, hogy a lemezmeghajtó kétoldalas-e, a program ezt nem veszi figyelembe.

Programcsomagunk részei a következők:

MS19GEN.BAS — BASIC program, amellyel a MS19SET.BIN gépi kódú programot állítjuk elő.

MSFORMAT.BAS — BASIC program a MS19SET.BIN futtatásához.

ADDLF.BAS — BASIC program, amely a CoCo szövegfájlokból hiányzó soremelő karaktert pótolja. Erre a legtöbb esetben (a szöveg megírásakor használt szövegszerkesztőtől függően) szükség van.

CoCo2MS.BAS — CoCo—MS—DOS konverter, melynek listáját is közöljük.

Lemezformálás

Ahhoz, hogy egy CoCo szövegfájlt IBM PC vagy más MS—DOS gépen olvasható lemezre tudjunk felvinni, először is egy MS—DOS igényű lemezt kell formattálnunk. Az MS—DOS lemezformátum a következő: 40 sáv, kilenc szektor/sáv, szektoronként 512 bájt. Ha a formattálást MS—DOS gépen végezzük a parancs a következő: FORMAT B: /1

(A RAINBOW cikk jelentős hányada eredetileg itt arról szól, hogyan állítsunk elő egy lemezen MS—DOS formátumot a CoCo-val. A megoldással még a CoCo-soknak sem érdemes foglalkozni, mivel a formattálás egy IBM PC-n egyszerűen elintézhető.)

Technikai fogások

Az IBM PC-kben alkalmazott NEC lemezkezelő chipnek van egy egyedi, sajátos tulajdonsága. Nevezetesen, valahányszor az index-lyukat érzékeli, ennek hardverreset a következménye. Ezt a CoCo-ban alkalmazott Western Digital lemezkezelő nem csinálja. Emiatt, ha a NEC chip által olvasandó lemezet készítünk a Western Digital chippel, az index-lyuk és az első szektorok fejléce között nagyobb távolságot kell tartani, mint amekkorát a Western Digital chip igényel. Ha ezt elmulasztjuk, a NEC chip az adott sáv első szektorát esetleg nem találja meg (attól függően, hogy az automatikus reset után feléled-e már ismét vagy sem). Ha egy szokásos CoCo formátumú lemezt kívánunk olvasni egy IBM PC-n vagy más, NEC chipet alkalmazó MS—DOS gépen, és az első szektort nem tudjuk kiolvastatni, próbálkozhatunk az index-

```
1 '*****************************
2 '*                           *
3 '*  COCO TEXTFILE MSDOS-RA   *
4 '*                           *
5 '*        THE RAINBOW        *
6 '*      1986.  JULIUS        *
7 '*                           *
8 '*****************************
9 CLEAR 4000,&H5DFF 'String terulet: 400
0 byte, BASIC programterulet $5DFF-ig
10 DIM NAM$(64),SG(64)
20 H=PEEK(&HC004):L=PEEK(&HC005):DKON=H*
256+L 'Disk I/O ROM szubrutin start cime
:C004 - C005
30 CG=0:MG=1 'Drive #0 & #1, CG=CoCo dis
k, MG=MS-DOS disk
100 CLS:PRINT"COCO MSDOS-RA"
110 PRINT:PRINT" (C) 1986. JAN.  MARTY G
OODMAN"
115 PRINT"HELYEZZE A COCO LEMEZT A DRIVE
0-BA, AZ MS-DOS LEMEZT A DRIVE1-BE"
120 PRINT"(AZ MS-DOS LEMEZ EGYOLDALASRA,
 9 SECTOR/TRACK-RE KELL LEGYEN FORMALVA)
"
140 PRINT:PRINT"NYOMJA MEG AZ <ENTER> GO
MBOT!"
150 IF INKEY$<>CHR$(13) THEN 150

200 'A kov. programszegmens beolvassa a
CoCo FAT-t, MS-DOS FAT-t, CoCo DIRECTORY
-t. Hiba eseten a program leall.
250 POKE &HEA,2:POKE &HE8,CG:POKE &HEC,1
7:POKE &HED,2:POKE &HEE,&H5F:POKE &HEF,0
255 EXEC DKON
260 IF PEEK(&HF0)<>0 THEN 9100
290 FOR N=2 TO 3
295 POKE &HE8,MG:POKE &HEC,0:POKE &HED,N
:POKE &HEE,&H72+(N-2)*2
297 EXEC DKON
260 IF PEEK(&HF0)<>0 THEN 9000
265 NEXT N
270 GOSUB 30000

300 'A kov. programszegmens kiirja a CoC
o DIRECTORY-t, hogy valaszthassunk egy F
ILE-t szam szerint. (A kivalasztott FILE
 szama az "A" valtozoban)
310 'Az eljarast nem kozoljuk - erdektel
en
330 PRINT"A VALASZTOTT FILE: ";NAM$(A)
340 PRINT"HA MEGFELEL, NYOMJA MEG AZ <EN
TER> GOMBOT!"
350 A$=INKEY$:IFA$=""  THEN 350
360 IF A$<>CHR$(13) THEN 300
```

```
1000 'FILE konverzio
1010 FZ=0
1015 YY=0
1020 G=SG(A)
1025 ZG=33
1030 'konverzio
1035 EF=0
1040 GOSUB 40000
1045 U=9
1050 IF G<70 THEN 1100
1060 U=G-100
1100 FOR J=1 TO U
1110 L=&H6000+(J-1)*512
1120 FZ=FZ+512
1121 PRINT@275,FZ
1122 IF G<68 THEN 1130
1125 IF J=U THEN EF=255
1130 GOSUB 25000
1135 IF YY=0 THEN WW=0C
1137 IF YY=0 THEN YY=255
1140 IF CXN=&HOAAA THEN 9300
1150 NEXT J
1160 IF EF<>255 THEN 1030
1170 GOTO 1030
2000 CLS:PRINT"ADJON NEVET AZ MS-DOS FIL
E-NAK' (MAX. 8 KARAKTER)"
2040 PRINT:INPUT A$
2050 IF LEN(A$)=0 OR LEN(A$)>8 THEN 2000
2060 B=LEN(A$)
2075 IF B=8 THEN 2095
2080 A$=A$+STRING$(8-B," ") ' Kiegeszul
az A$ string 8 karakter hosszusagura
2095 A$=A$+"COL"
2100 'DIRECTORY AZ MS-DOS LEMEZRE
2105 GOSUB 20000
2110 IF A$<>"" THEN 2120
2112 CLS:PRINT"AZ MS-DOS LEMEZ MEGTELT.
TEGYEN EGY UJABB MS-DOS LEMEZT A DRIVE#1
-BE, MAJD NYOMJA MEG AZ <ENTER> GOMBOT!"
2114 IF INKEY$="" THEN 2114 ELSE 100
2120 POKE &HEA,3:POKE &HE8,MG
2130 'FAT IRAS AZ MS-DOS LEMEZRE
2140 FOR N=2 TO 5
2145 LO=&H72+(N-2*INT(N/2))*2
2150 POKE &HED,N:POKE &HEE,LO
2160 EXEC DKON
2170 NEXT N
2180 GOTO 300


9000 CLS:PRINT"ROSSZ MS-DOS LEMEZ":END


9100 CLS:PRINT"ROSSZ COCO LEMEZ":END
```

a μM-ban

```
15000 'FAT OLVASAS
15001 'GN=CLUSTER (0-400)
15003 'CV=CLUSTER TARTALOM
15005 FB=&H7200
15010 G1N=INT(GN/2)
15020 GCN=3*G1N
15030 GF=GN-2*G1N
15040 B1=PEEK(FB+GCN)
15050 B2=PEEK(FB+GCN+1)
15055 B3=PEEK(FB+GCN+2)
15060 N1=(B1 AND &HF0)/16
15070 N3=(B2 AND &HF0)/16
15080 N5=(B3 AND &HF0)/16
15090 N2=B1 AND &H0F
15100 N4=B2 AND &H0F
15110 N6=B3 AND &H0F
15120 IF GF=0 THEN 15200
15130 CV=N3+N6*16+N5*256:RETURN
15200 CV=N2+N1*16+N4*256:RETURN

17000 'ERTEKADAS EGY ADOTT CLUSTERNEK
17001 'GN=CLUSTER SZAM
17002 'CV=12 BIT-ES ERTEK A GN-BE
17003 'MINDEZ A "FAT BUFFER" TERULETEN
17010 N1=INT(CV/256):T=CV-256*N1
17020 N2=INT(T/16)
17030 N3=T-16*N2
17050 G1N=INT(GN/2)
17060 GCN=3*G1N
17070 GF=GN-2*G1N
17075 GOSUB 17300
17080 IF GF=0 THEN 17200
17100 'IRAS A PARATLAN SZ. "CLUSTER"-BE
17110 B2=B2 AND &HOF:B2=B2+16*N3
17120 B3=16*N1+N2
17130 POKE &H7200+GCN+1,B2
17140 POKE &H7200+GCN+2,B3
17150 RETURN
17200 'IRAS A PAROS SZ. "CLUSTER"-BE
17210 B1=16*N2+N3
17220 B2=B2 AND &HF0:B2=B2+N1
17230 POKE &H7200+GCN,B1
17240 POKE &H7200+GCN+1,B2
17250 RETURN
17300 B1=PEEK(&H7200+GCN)
17310 B2=PEEK(&H7200+GCN+1)
17320 B3=PEEK(&H7200+GCN+2)
17340 RETURN

20000 'MS-DOS DIRECTORY-BAN SZABAD HELYE
T KERES. HA VAN, "A$"+"ATTRIBUTE BYTE"-O
T BEIRJA (ATTR. BYTE=2)
20002 'FILE-HOSSZAT NEM IRJA BE
20003 'GYAKORLATILAG MEGFELEL AZ "OPEN"
PARANCSNAK
20004 'HA A$="", A DIRECTORY MEGTELT (A$
 HOSSZA: 11 BYTE)
20007 'WW=1. CLUSTER
20010 A$=A$+CHR$(&H20)+STRING$(14,CHR$(0
))
20020 X=WW:GOSUB 45000
20030 G1$=CHR$(L):D2$=CHR$(H)
20035 GOSUB 46000
20040 A$=A$+G1$+G2$+CHR$(D1)+CHR$(D2)+CH
R$(D3)+CHR$(0)
20045 GOSUB 20100
20050 IF A$="" THEN RETURN
20060 FOR Q=0 TO 31
20062 POKE MDB+K*32+Q,ASC(MID$(A$,Q+1,1)
) 'A$ Q+1-EDIK KARAKTERE
20064 NEXT Q
20070 POKE &HEA,3:POKE &HEB,MQ
20075 EXEC DKON
20080 RETURN
20100 FOR N=0 TO 3
20110 MDB=&H7600:MSB=&H76
20120 POKE &HEA,2:POKE &HEB,MQ:POKE &HEC
,0:POKE &HED,6+N:POKE &HEE,MSB:POKE &HEF
,0
20130 EXEC DKON
20135 IF PEEK(&HF0)<>0 THEN 9100
20140 FOR K=0 TO 15
20150 FB=PEEK(MDB+32*K)
20160 IF FB=&HE5 THEN RETURN
20170 IF FB=0 THEN RETURN
20180 NEXT K
20190 NEXT N
20195 A$="":RETURN
25000 'SZABAD "CLUSTER"-T KERES, MAJD AZ
 L SZ. SZEKTORBAN A SZABAD CLUSTER-BE IR
25002 'AZUTAN MEGKERESI A KOV. SZABAD SZ
EKTORT, A "FAT"-T RENDEZI, A KOV. SZABAD
 CLUSTER-T MEGJEGYZI
25005 'CXN=A KOV. SZABAD CLUSTER. HA NIN
CS SZABAD, CXN=&HOAAA
25007 'HA EF=255, NEM KERES TOVABB SZABA
D CLUSTER-T ES A "FAT"-BE BEIRJA EZT AZ
ALLAPOTOT
25010 Q=INT(L/256):QL=L-256*Q
25040 IF ZQ=33 THEN CXN=2:ZQ=0
25050 FOR GN=CXN TO 352
25060 GOSUB 15000
25070 IF CV=0 THEN 25100
25080 NEXT GN
25090 CXN=&HOAAA:RETURN
25100 'MEGKERESI A KOV. SZABAD "GRANULE"-
T
25105 OC=GN
25110 IF EF=255 THEN 25200
25120 FOR GN=OC+1 TO 352
25130 GOSUB 15000
25140 IF CV=0 THEN 25200
25150 IF CV=&HOFF7 THEN 2520
25160 NEXT GN
25170 CXN=&HOAAA:RETURN
25200 CXN=GN
25210 GN=OC:CV=CXN
25215 IF EF=255 THEN CV=&HOFFF
25220 GOSUB 17000 '"FAT" CIME A BUFFERBE
25230 'KISZAMALJA OC-BOL A TRACK-ET ES S
ZEKTORT
25240 T=INT((OC-2)/9)+1:S=OC+7-9*T+1
25250 POKE &HEA,3:POKE &HEB,MQ:POKE &HEC
,T:POKE &HED,S:POKE &HEE,Q:POKE &HEF,QL
25260 EXEC DKON
25265 POKE &HFF40,8
25270 RETURN

30000 ' BEOLVASSA A COCO DIRECTORY-T
30001 'NAM$(N)=FILE
30002 'SG(N)=1. GRANULE
30005 'LE=UTOLSO FILE
30010 K=-1:N=1:LE=0
3
30020 'HUROK
30030 K=K+1
30040 S=INT(K/8)+3
30045 KS=K-8*INT(K/8)
30050 DSKI$ 0,17,S,A$,B$ 'A "DSKI$" PARA
NCS BEOLVAS EGY SZEKTOR-TARTALMAT (JELEN
LEG A DRIVE #0-ROL, 17. TRACK, S SZEKTOR
, ELSO 128 BYTE AZ "A$"-BA, A MASODIK 12
8 BYTE A "B$"-BA)
30060 C$=A$
30070 IF KS>3 THEN C$=B$
30080 KS=KS-(4*INT(KS/4))
30090 T$=MID$(C$,32*KS+1,32*(KS+1))
30100 FB=ASC(T$)
30110 IF FB=0 THEN 30030 'LETOROLT FILE
30120 IF FB=255 THEN RETURN 'NINCS TOBB
FILE
30140 NAM$(N)=LEFT$(T$,8)
30150 NAM$(N)=NAM$(N)+"."
30160 NAM$(N)=NAM$(N)+MID$(T$(9,3)
30170 SG(N)=ASC(MID$(T$,14,1))
30180 LE=N:N=N+1:GOTO 30030
40000 '2 "GRANULE" AZ ADATBUFFERBE
40001 'G71. GRANULE
40003 'HA NINCS TOBB, G=100+A BUFFERBEN
LEVO 512 BYTE-OS SZEKTOROK SZAMA
40010 DF=0 '1. GRANULE
40020 'HUROK
40025 IF DF>1 THEN RETURN
40030 DB=&H6000:IF DF<>0 THEN DB=&H6900
40040 GT=G:IF GT>33 THEN GT=GT+2
40050 T=INT(GT/2):D=GT-2*T
40055 S=1:IF D<>0 THEN S=10
40060 GOSUB 41000
40100 G=PEEK(&H5F00+G) 'KOV. GRANULE #
40110 DF=DF+1
40120 IF G<&H69 THEN 40020
40130 G=G-&HC0
40140 IF G>9 THEN 9100
40160 G=G+1
40170 IF DF?2 THEN G=G+9
40180 G=INT(G/2):G=G+100
40190 RETURN
41000 'A "GRANULE"-T A BUFFERBE IRJA
41010 FOR N=0 TO 8
41020 POKE &HEA,2:POKE &HEB,CD:POKE &HEC
,T:POKE &HED,S+N:POKE &HEE,DB/256+N:POKE
 &HEF,0
41030 EXEC DKON
41035 IF PEEK(&HF0)<>0 THEN 9100
41040 NEXT N
41050 RETURN
45000 H=INT(X/256):L=X-256*H:RETURN
46000 D3=INT(FZ/65536)
46010 FZ=FZ-65536*D3
46020 D2=INT(FZ/256)
46030 D1=FZ-D2*256
46040 RETURN
```

lyuk leragasztásával. A közönséges szektorolvasáskor az index-lyuknak amúgy sincs szerepe, így letakarása a lemezkezelőt a lemez olvasásában nem befolyásolja károsan.

A soremelés problémája

Bár a program a CoCo szövegfájl-ok (vagy más egyéb fájlok) valamennyi bájtját helyesen írja át MS—DOS fájl formába, ez már nem elegendő. Az előzőkben már beszéltünk arról, hogy az MS—DOS típusú fájlokban minden 'kocsi vissza' (CR=Carriage Return) vezérlőkarakter után 'soremelő' (LF=Line Feed) áll. Ezért szükségünk van egy külön programra, hogy a fordítás után kihúzzuk a fájlból a LF karaktereket, hogy a CoCo-n futó szövegszerkesztőket ne zavarják (amely szövegszerkesztők nemcsak hogy nem használják a LF karaktert, de ki is 'akadnak', ha találkoznak vele).

A másik irányba konvertáláskor (csaknem minden esetben) be kell szúrnunk egy LF-et valamennyi CR után. Enélkül a konvertált fájl továbbra is olvashatatlan marad a legtöbb MS—DOS-ban futó szövegszerkesztő számára. Mielőtt tehát MS—DOS lemezre kerülne a CoCo szövegfájl, egy LF-ekkel ellátott fájlt készítünk belőle.

Konvertálás

Eljutottunk odáig, hogy betölthetjük és futtathatjuk a konvertáló programot, mely menüre szervezett. Tegyük a CoCo Disk BASIC típusú lemezt a Drive 1-be. Az ENTER gomb lenyomása után a program ellenőrzi, hogy helyesen jártunk-e el. Elolvassa ezután a lemez Directory-ját és megjeleníti a képernyőn, hogy válasszuk ki a konvertálandó fájlt.

Végül meg kell adni a fájl nevét, amellyel felkerül az MS—DOS lemezre. Ezután vagy kiszállunk a programból, vagy újra indítunk. A konvertálás ideje a fájl hosszának függvénye, de meglehetősen gyors (2400 Baud).

Emlékeztetőül: valamennyi konvertált szöveg a rootdirectory-ba kerül. Nem alkalmas ui. a program arra, hogy subdirectory-t állítson fel és abba írjon. Továbbá: az értékes rész végére általában 'szemét' is kerül. A 'szemét' rendszerint magából a fájlból származik (lásd az előző cikket!).

A programot más típusú gépekre adaptálni kívánó olvasóinknak a szerkesztőség további információkkal szolgálhat.

Vágh István

Mesterséges értelem

A Földön és — ismereteink mai állása szerint — még sok fényévnyi távolságon belül az emberi agy a létező legbonyolultabb dolog. Az ember az előtte álló feladatok egyik legnagyobbikával bajlódik, amikor saját agyát, saját gondolkozási mechanizmusait akarja megérteni, utánozni. A mesterséges intelligencia területén munkálkodók ezzel próbálkoznak, számítástechnikai módszerek felhasználásával.

A számítógépek megalkotása az ember számára nemcsak azt a többletet adta meg, hogy számításokat gyorsabban végezhet el, mint ezt a saját agyával, saját kezével tehetné, hanem egy ennél sokkal nagyobb horderejű lehetőséget teremtett. Arról van szó, hogy egy digitális számítógéppel végtelen sok másik számítógép, számítási folyamat működését megvizsgálhatjuk. Elvileg nincs akadálya tetszőleges matematikai vagy fizikai rendszer — definiálása után aktuális — kísérleti megfigyelésének a működés számítógépes szimulációja segítségével. Feltételezve, hogy az emberi agy működése, bármilyen bonyolult is legyen, mégis valamilyen fizikai, számítási folyamat, ennek számítógépes modellezése elvileg szintén reálissá vált.

Napjainkban az agytevékenység megértésétől, vagyis — számítástechnikai alapfogalmakkal kifejezve — az emberi intelligencia szoftverjének „elkülönítésétől, leemelésétől" az agy neuronhardverjéről még nagyon távol állunk, de a próbálkozásokból gyakorlati alkalmazási lehetőségekkel bíró módszerek születtek.

Bármilyen összetett szerkezet, így az agy folyamatait is két irányból elindulva kísérelhetjük meg megérteni. Megpróbálhatjuk az alkotórészek működésének eredőjeként felfogni, vagy a látható jelekből, felismerhető szabályosságokból kikövetkeztetni a működést. Kezdetben az első módszer volt előtérben. Az idegsejt egy egyszerű modelljét, a perceptront akarták felhasználni arra, hogy belőle összetett, sokszínű, például a látókéreg felépítéséhez hasonló hálózatot felépítve megértsék az agyműködést. Rövidesen bebizonyosodott, hogy ez az elképzelés túl egyszerű, és csak néhány alacsonyabb szintű működés tanulmányozására megfelelő. Bár a kutatások ebből az irányból tovább folynak azóta is — jóval bonyolultabb modellek felhasználásával —, a hangsúly a másik megközelítésre tevődött át.

Így a későbbiekben a mesterséges intelligencia kutatások keretében olyan programok létrehozásán dolgoztak, amelyek egy-egy, az emberi intelligencia részének tartott képesség reprodukálására alkalmasak. Ilyenek voltak a különböző játékok, a matematikai tételek bizonyításai, a szóbeli utasítások megértése, egyszerű beszélgetés folytatása, az alakzatok felismerése, képek értelmezése. Ezen emberinek tartott kvalitások közül néhánynak a leutánzása egészen látványosan sikerült a számítógépekkel. Az elkészült programok esetenként meghaladták egyes állatok idegrendszerének nívóját vagy — bár rendkívül szűk területen — szinte emberi viselkedést mutattak. De még nagyon távol állunk attól, hogy az emberi agy rugalmasságát, teljesítőképességét, nagy adattömeg összefüggéseit felismerő és felhasználó képességét elérjük.

Ma a harmadik korszakot éljük, amikor már — a súlyos elvi nehézségek ellenére — az eddigi vizsgálódások során létrehozott módszerek gyakorlati alkalmazásait keresik. Ezt magyarázza, hogy a mikroprocesszorok megjelenése és a RAM áramkörök árának lezuhanása olyan bonyolultságú alkalmazások előtt nyitott utat, amelyekben már érdemes ezeket a (nagy számítástechnikai teljesítményigényük miatt) eddig gyakorlati célokra nem alkalmazott módszereket is felhasználni.

Az elvi problémákat nem elemezve, kizárólag az alkalmazásokban szerepet játszó részek kifejtésével szeretnék ezek közül néhány érdekesebbet bemutatni.

A számítógépes problémamegoldás alapjai

Elsőként foglalkozzunk az intelligens viselkedés egyik nagyon fontos meghatározójával, a problémamegoldási képességgel. Érthető módon, mivel ez a képesség az emberi létnek is meghatározó alkotóeleme, a gépi problémamegoldásnak nagyon széles gyakorlati alkalmazási területe van.

Mindeddig a számítógépek csak támogatták az embert a különböző feladatok megoldásában: segítették visszakeresni a szükséges adatokat és gyorsan elvégezni a kijelölt számításokat. Magának a problémának az elemzése, a célravezető utak kijelölése végül is mindig az ember feladata maradt. Ezen változtat a szoftvertermékek új nemzedéke. A szóban forgó programok a problémamegoldási feladatok egy részét át fogják venni az embertől, illetve támogatják olyan területeken, mint például az orvosi diagnosztika, ahol nem lenne „egészséges dolog" önállóan a gépre bízni a döntést.

Formális problémadefiníció

Ahhoz, hogy képessé tegyük a számítógépet problémamegoldásra, először is formálisan definiálnunk kell számára a problémát. Nézzünk meg egy egyszerű példát.

Adott két, kezdetben üres edény, egy 4 literes és egy 3 literes.

Van egy, vizet korlátlan mennyiségben tartalmazó kút, amelyből bármelyik edényt teletölthetjük.

Szabad a vizet az egyik edényből a másikba áttölteni vagy a földre kiönteni.

Feladat: érjük el azt az állapotot, hogy a 4 literes edényben 2 liter víz legyen.

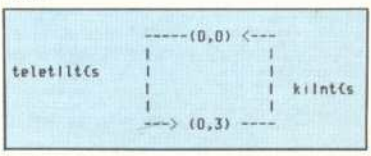
Az első részfeladat, hogy valamilyen módon nyilvántartsuk a helyzetet. A tárgyalt egyszerű esetre az egyik lehetséges eljárás, hogy egy kételemű vektor (tömb) első eleme jelölje a 4 literes edényben levő vízmennyiséget, a második eleme pedig a 3 literes edényben levő vízmennyiséget. Ebben a leírásban a kiindulási állapot (0,0). A célállapot, amit a probléma megoldásának tekintünk, a (2,0) állapot. A lehetséges állapotok (az első pozícióban 0—4 számok, a másodikban 0—3 számok tetszőleges kombinációja állhat) összességét *állapottér*nek vagy *problématér*nek szokás nevezni. Jelen problématérben kitüntetett szerepe van egy kiindulási (0,0) és egy végállapotnak (2,0).

Hogyan oldaná meg egy ember ezt a problémát? A lehetséges cselekvéseket és következményeiket agyában lejátszva olyan cselekvéssort keresne, amelynek eredményeképpen a kívánt helyzet kialakul. Matematikai megfogalmazásban: vannak megengedett műveleteink, ezek az állapottér egyik pontjából a másikba visznek át bennünket, és egymás utáni alkalmazásukkal szeretnénk egy olyan sorozatot összeállítani, amely az állapottér kiindulási pontjából a célpontba visz. Megengedhető műveletek például:

— Ha a 3 literes edény üres, töltsd tele vízzel.

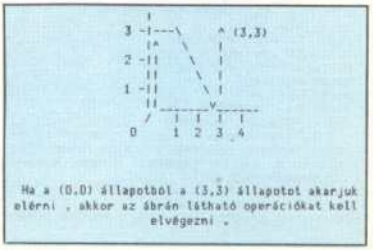
— Ha a 4 literes edény üres, töltsd belé át a 3 literes edény tartalmát.

Általánosságban egy szabály formája olyan, hogy az elején szereplő feltétel teljesülése esetén alkalmazható, a másik oldalán pedig egy cselekvés vagy cselekvéssorozat található. Ezzel el is jutottunk a problémák formális definíciójához. A problémát leírja:



1. ábra

2. ábra
Ha a (0,0) állapotból a (3,3) állapotot akarjuk elérni, akkor ezeket az operációkat kell elvégezni



VÖDÖRBŐL CSÖBÖRBE, avagy csöbörből vödörbe

— egy állapottér;
— az állapottér bizonyos kitüntetett pontjai, ezek a kiindulási és a végpontok;
— egy szabálygyűjtemény, amely az egyik állapotból az állapottér valamely másik pontjába való eljutás szabályait rögzíti.

Ahhoz, hogy a számítógépes problémamegoldás legalapvetőbb fogalmához, a *produkciós rendszer*hez eljussunk, az eddig említettekhez még egy dolgot hozzá kell tennünk. Ez a döntési stratégia, amely — olyan esetekben, amikor több szabály alkalmazhatósági feltétele is teljesül — segít a megfelelő lépés kiválasztásában. A stratégiával szemben az első követelmény, hogy okozzon változást. Megtehetnénk, hogy először teletöltjük a 3 literes edényt vízzel, aztán kiöntjük a földre. Nyilvánvaló, hogy ez esetben körben járunk az állapottérben. Következésképpen semmilyen stratégia nem megfelelő, amelyik akár több állomással is, de körbevisz az állapottérben. *(1. ábra)*

A második követelmény, hogy szisztematikus legyen, hiszen matematikailag lehetséges, hogy a szabályok véletlen egymás utáni alkalmazásával soha nem jutunk el a célállapotba. Tehát a stratégiát követve tetszőleges sorrendben, de el kell jutnunk az állapottér mindegyik pontjába, ha az addig bejárt pontok egyike sem volt célállapot. *(2. ábra)*

Feladat: készítsünk a 2. ábrához hasonló ábrát, amely az eredeti feladatban szereplő (2,0) célállapotba vezető utat adja meg.

A produkciós rendszer szerkezetű program elemei

Egy program, ha megadott szabályok alkalmazásával kapott állapotokat követ, leginkább egy interpreterre hasonlít, amely veszi a következő alkalmazandó szabályt, majd az e szerinti eljárás eredményét feljegyzi a rendszer állapotát leíró adatstruktúrába. Ha egy feladathoz ilyen, tehát ún. produkciós rendszer szerkezetű megoldóprogramot akarunk készíteni, számos dolgot alaposan át kell gondolnunk. Ezek közül néhányról lesz most szó.

A kiindulási és a célállapotot — mindkettőből lehet több is — összekötő utak keresése kétféle közelítés szerint haladhat: előre a kiindulási állapotokból, illetve visszafelé a célállapotoktól. A kedvezőbb irányt két tényező határozza meg. Az egyik, hogy a kiindulási vagy a célállapotokból van-e több. A problématérben bolyongva nyilván könnyebben akadunk olyan állapothoz tartozó pontba, amelyből több van. A hatékonyabb keresési irány másik meghatározója az elágazási faktorok viszonya, vagyis az, hogy melyik irányba haladva ágazik el gyakrabban a keresési fa. Abba az irányba érdemesebb haladni, amerre kisebb az elágazások száma és így az eltévedés lehetősége.

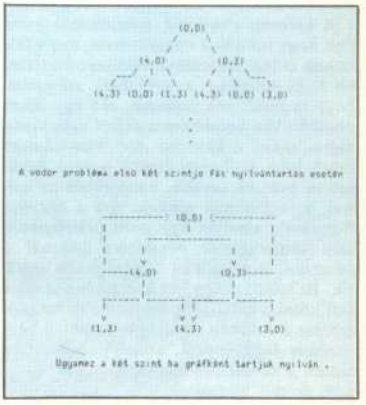
A másik megfontolandó kérdés az állapottér nyilvántartásának módszere. Mégpedig az, hogy faként vagy gráfként tartjuk nyilván, azaz hogyan cselekszünk abban az esetben, ha valamilyen operáció végrehajtása visszavisz bennünket a tér egy olyan pontjába, ahol már jártunk. Nyilvántartjuk-e egyáltalán azt, hogy hol jártunk már, és ellenőrizzük-e azt, hogy hova érkeztünk? Ha nem ellenőrizzük, ez nyilván lerövidíti egy adott pont feldolgozási idejét, ugyanakkor nem vesszük észre, hogy esetleg többször elérkezünk egy pontba, és minden alkalommal megvizsgáljuk az onnan induló utakat. A nyilvántartás módját végül is az állapottér szerkezete határozza meg: az, hogy átlagosan hány operáció juttat vissza egy már érintett csomópontba, ezek száma hogyan függ az ág mindenkori mélységétől, valamint hogy egyáltalán mekkora a problématér. *(3. ábra)*

Feladat: képzeljünk el egy olyan helyzetet, amelyben minden pont átlag *n* alkalommal generálódik, és programunk számára ugyanannyi időt vesz igénybe ellenőrizni, hogy egy csomópontot már érintettünk-e, mint *m* db újabbat generálni. Mikor jut a program gyorsabban a megoldáshoz: ha faként, vagy ha gráfként kezeljük a problémát?

A következő kérdés akkor merül fel, ha az állapottér egyes pontjai nem ilyen kevés változóval, hanem nagyon sokkal írhatók le, ami a bemutató példák kivételével többnyire így is van. Némely problémánál komoly gondot okozna, ha a keresési gráf minden pontjában tárolni szeretnénk a teljes állapotleírást. Ennek elkerülésére az ad lehetőséget, hogy minden egyes transzformációs szabály valószínűleg csak kevés állapothatározót módosít, ekkor pedig mgpróbálkozhatunk azzal, hogy csak egy helyen tartjuk nyilván a teljes állapotleírást. De hol legyen ez a hely? Ha az állapotleírást mindig az éppen vizsgált csomópontban tartjuk, akkor — ha rossz irányba haladtunk, ami elég gyakori eset, és zsákutcába kerülve nem jutottunk el a megoldáshoz — vissza kell térnünk egy korábbi állapothoz, és helyreállítva az ott érvényes állapotot, újra kell próbálkozni. Ehhez azonban ellenkező irányban is érvényesülniük kell a transzformációknak, amelyek visszavezetnek az elágazási pontig. Viszont ha csak a kiindulási állapotot tartjuk nyilván teljes egészében, és minden pontban csak az elvégzett transzformációt, akkor a csomópontokban előálló helyzet értékelése válik nehézkessé.

További feladat az alkalmazható szabályok felismerése bonyolult állapotleírás esetén. Itt tulajdonképpen két nehézség van. Egyrészt nem biztos, hogy könnyen megállapítható egy adott szabály feltételeiről, teljesülnek-e, másrészt nagyon számításigényes minden lépésnél egyenként megvizsgálni az összes szabályra a feltételek teljesülését. Egy szabály teljesülésének feltételeit nehéz lehet ellenőrizni részint azért, mert nehéz az illesztés az állapotleírás összetettségéből adódóan, részint azért, mert a szabályok megfogalmazása olyan általános, hogy a teszt valójában számítások végzését, változók behelyettesítését igényli.

Könnyen illeszthető specifikus szabály a vödör-problémához: HA az állapot (0,2), AKKOR öntsd át a 3 literes edény tartalmát a 4 literesbe. Változóbehelyettesítés és teszt szükségessége miatt nehezebben il-



3. ábra
A vödör-probléma első két szintje fás nyilvántartás esetén

Ugyanez a két szint, ha gráfként tartjuk nyilván

leszthető általános szabály: HA az állapot (a,2) és a $\neq 0$, AKKOR üritsd ki a 4 literes edényt.

Külön gondot jelent ilyen esetben az is, ha egy szabály többféleképpen illeszthető az adott állapotra. Ez olyankor fordul elő, ha többféle változóérték-helyettesítéssel is kielégíthető a feltétel. Elképzelhető, hogy nem szükséges az alkalmazandó szabály pontos illesztése, elég, ha megközelítő egyezést kapunk. Ilyenkor egy szabály jóval több esetben választódik ki alkalmazásra. Ez akkor szükséges, ha azt akarjuk, hogy egy adott pontban lehetőleg legyen alkalmazható szabályunk, ne álljon elő az az eset, hogy egyik szabály alkalmazhatósági feltétele sem teljesül. Több szabály alkal-

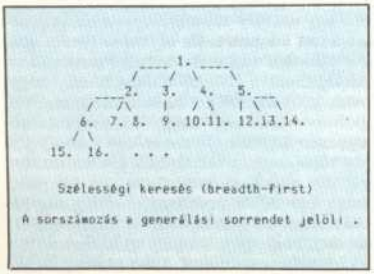
mazhatósága esetén az illesztés során célszerű figyelembe venni, hogy ha a teljesülő szabályok között van olyan, amelyik valamely másiknak speciális esete, akkor inkább a speciális szabályt alkalmazzuk, hiszen az valószínűleg inkább követendő cselekvést ad meg. Minél specifikusabb szabályokat veszünk bele a program szabálybázisába, annál több munkát végzünk el magunk, és kevesebb kikövetkeztetni valót hagyunk a gépre.

Miután minden pontban kiválasztjuk az alkalmazható szabályokat, a döntés, hogy ezek közül végül melyiket vegyük a továbblépéshez, a felhasznált stratégián múlik. El kell határoznunk, hogy az értékelés, amit azért teszünk, hogy a következő szabályt kiválasszuk, mennyire legyen körültekintő. Ha bonyolult, azaz számításigényes megfontolásokat alkalmazunk, akkor egy lépés értékelése viszonylag sokáig tart, de biztosabban vezet a célállapotok felé. Egyszerű értékelőfüggvénnyel viszonylag gyorsan sok csomópontot járunk be, de ezalatt nagyobb esélyünk van rá, hogy rossz irányba haladunk. Tekintsük át röviden a lényegesebb keresési stratégiákat.

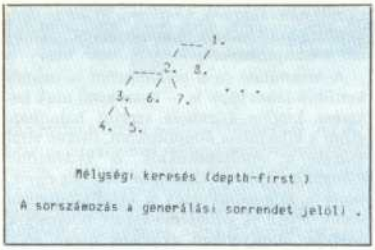
Keresési stratégiák

A keresési stratégiák feloszthatók aszerint, hogy tudunk-e valamit arról, merrefelé jutunk el leghamarabb a célállapotba, illetve hogy megéri-e a szükséges számítást, hogy tudjunk róla valamit. Ha egy adott pontban van becslésem a céltól való távolságra, akkor a keresést már irányíthatom azzal, hogy az ismert értékű pontok közül onnan keresek tovább, amelyikből az esélyek a legjobbnak tűnnek. Azt a közelítő függvényt, amelyet egy pont célállapottól való távolságának becslésére használ a program, heurisztikus függvénynek nevezik. Itt tulajdonképpen kompromisszumot kell kötni a célállapot leggyorsabb megközelítése érdekében. Minél pontosabb a heurisztikus függvényem, annál pontosabban tudom a követendő irányt, de annál kevesebb pontot vizsgálhatok meg ugyanannyi idő alatt.

Ha úgy határozunk, hogy nem éri meg a szükséges számításokat, vagy esetleg nem is tudunk módszert adni az adott pont célállapottól való távolságának meghatározására, akkor az úgynevezett tájékozatlan keresési stratégiák valamelyikét kell alkalmaz-



4. ábra
Szélességi keresés. A sorszámozás a generálási sorrendet jelöli



5. ábra
Mélységi keresés. A sorszámozás a generálási sorrendet jelöli

nunk. Lényegében két ilyen stratégiát különböztetünk meg: a szélességi (breadth-first) és a mélységi (depth-first) keresést. A szélességi keresés során a szerint a rendszer szerint haladunk, hogy a kiindulási állapottól adott, egyenlő távolságra levő pontok mindegyikéhez előállítjuk az onnan elérhető állapotok mindegyikét, ezzel egy újabb, eggyel távolabbi szintet létrehozva, ahol aztán ismét ilyen módon lépünk tovább. Az egymás alatti szintek generálását addig folytatjuk, amíg a minden újonnan generált pontra végrehajtott teszt azt nem jelzi, hogy célállapotba jutottunk. *(4. ábra)*

A mélységi keresés ehhez igen hasonló azzal a különbséggel, hogy itt először egy ágon olyan messzire elmegyünk, amennyire csak lehetséges. Új ágra akkor térünk át, ha bebizonyosodott, hogy zsákutcába kerültünk. *(5. ábra)*

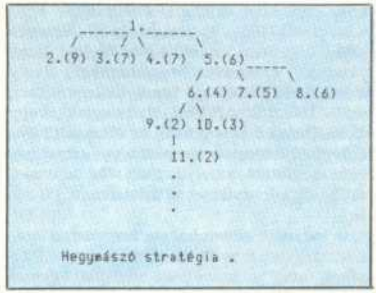
Meddig tart a vízszintes stratégia?

Ha úgy ítéljük, hogy egy kisebb mennyiségű számítás végzése indokolt, kiszámíthatjuk egy pont összes leszármazottjainak a távolságértékét, azaz a heurisztikus függvény értékét azokban a pontokban, és a legkedvezőbbet választjuk kiterjesztésre. Ezt hegymászás-stratégiának hívják. A hasonlat alapja, hogy a hegymászó ködben haladva a csúcs felé, mindig a tartózkodási pontjához képest legmeredekebb úton mászik felfelé. Választott stratégiánk félre is vezethet: a hegycsúcsot körülvevő dombok egyikére felérve lokális maximumpontban találhatjuk magunkat, ahonnan minden út lefelé visz, azaz bármerre haladunk tovább, távolodunk a célállapottól. Hasonló gondot jelent a fennsík, amikor egyszerre csak azt tapasztaljuk, hogy bármerre nézünk, a hegyoldal gradiense, emelkedése ugyanakkora. *(6. ábra)*

Ha még ennél is több számítási munkát tartunk szükségesnek a következő lépés kiválasztásához, akkor megtehetjük, hogy minden eddig generált, de ki nem terjesztett pontra kapott értéket figyelembe veszünk a következő vizsgálandó pont kiválasztásához. Ennek a nevét (best-first) talán „rögtön a legjobbat"-nak fordíthatnánk.

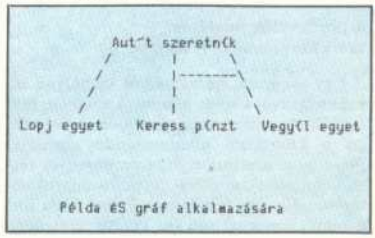
Mindeddig csak VAGY-gráfokkal foglalkoztunk, de amikor egy probléma megoldását részproblémákra bontás segítségével akarjuk meggyorsítani, figyelembe kell vennünk, hogy a problémát csak akkor tekinthetjük megoldottnak, ha minden részprobléma meg van oldva. Ilyenkor ÉS-gráfokkal kell dolgoznunk. Az ÉS-gráfok esetén nemcsak egy, hanem két vagy több ágon kell eljutnunk a megfelelő részproblémák megoldásához. *(7. ábra)*

Eddig csak olyan algoritmusokat említettem, amelyek egy előre eldöntött irányba haladnak. Elindulunk a cél- vagy a kiindulási állapotból, és addig keresünk, amíg a másik véget jelentő valamelyik állapotok egyikébe eljutunk. Lehetséges az is, hogy mind a két irányt egyszerre vizsgáljuk. Ösz-



6. ábra
Hegymászó stratégia. A sorszámok a generálás sorrendjét adják meg, a pontok melletti számok a pont heurisztikus függvény által becsült távolságát a céltól

7. ábra
Példa ÉS-gráf alkalmazására. A 2. és 3. ágat egy időben kell teljesíteni az óhajtott cél eléréséhez. (Talán ezért is választják néhányan az első lehetőséget . . .)

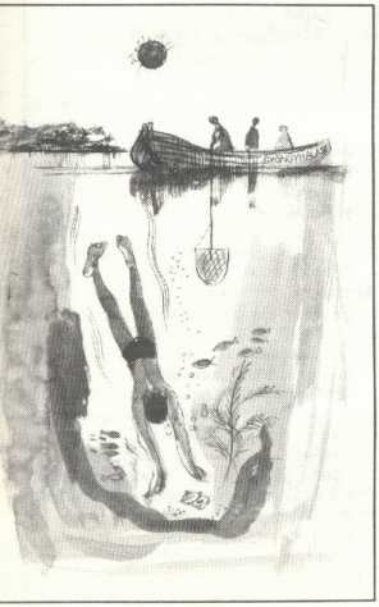


Kétoldali megközelítés a „problématérben"

szegyűjtjük az alkalmazható műveleteket mind a két irányból. Egy megfelelő függvény segítségével értékeljük, hogy ezek közül melyik csökkenti legjobban a két állapot közötti különbséget. Ennek végrehajtása után újra megismételjük a fenti eljárást az így kapott állapotokra. Ezt addig ismételjük, amíg a két út összeér.

Bár már most nyilvánvaló, hogy a produkciós rendszermegközelítés alkalmazása során számos nehéz döntés elé kerülünk,

Merülő statégia



de ha ezeket helyesen hozzuk meg, akkor a hagyományos módszerek által lehetővé tett megközelítésnél hatékonyabb és sokkal rugalmasabb módszert kapunk a problémamegoldási feladatok programozásához. A rugalmasság az ismeretek és a vezérlési struktúra jobb elkülönüléséből adódik. A szabálybázis jó áttekinthetősége, módosíthatósága és bővíthetősége megkönnyíti az ezzel a módszerrel készült programok későbbi igényekhez való hozzáalakítását. Ezek azok az okok, amelyek mostanában ráirányították a figyelmet ezekre a módszerekre.

Feladat: készítsünk programot egy majom vezérlésére. A majom egy szoba bejáratánál áll. A szoba közepén banán lóg a plafonról. A majom éhes, és meg akarja enni a banánt, de a padlón állva nem éri el. A szoba ablakánál van egy doboz, amit a majom felhasználhat. A majom a következő cselekvésekre képes: odamenni valahova; felmászni a dobozra; eltolni a dobozt valahova; megfogni a banánt, ha közvetlenül alatta a dobozon áll.

NP-teljesség és heurisztika

Napjainkban, amikor a rendelkezésre álló számítástechnikai teljesítmények hihetetlen gyorsasággal nőnek, felmerülhet a kérdés, hogy érdemes-e bonyolult megoldásokat keresni programok hatékonyságának növelésére, hiszen rövid időn belül úgyis készül majd olyan számítógép, amely elég nagy és gyors lesz ahhoz, hogy az adott feladatot elfogadható időn belül megoldja. Létezik azonban a problémáknak egy olyan fajtája, amelyet NP-nehéz és egy másik, amelyet NP-teljes problémáknak neveznek. Ilyen például az utazó üzletember példája:

Adott n db város, és ezek távolsága egymástól ($(n(n-1))/2$ db távolságérték). Keressük azt a legrövidebb utat, amely pontosan egyszer érint minden várost; azaz azt az utat, amelyet egy olyan üzletembernek kell bejárnia, akinek az a feladata, hogy menjen el minden városba, de ezt összességében a legkisebb útiköltséggel tegye.

Ha az optimális megoldást keressük, meg kell vizsgálnunk minden lehetséges útvonalat. n város esetén ez $(n-1)!$ különböző utat jelent. Egy út megvizsgálási ideje arányos az érintett városok számával, azaz n-nel, vagyis a keresés teljes időtartama $n!$-sal arányos.

Szükséges és elégséges feltétel

Ez már 10 város esetén is 3 628 800 út. A számítástechnikában és különösen azokban az esetekben, amikor a számítógéptől valamilyen tekintetben intelligens viselkedést várunk, igen gyakran kerülünk szembe ehhez hasonló problémákkal. Ráadásul ezekben az esetekben a városok számának megfelelő adatok száma százakkal, de inkább ezrekkel fejezhető ki. Röviden: napjaink leggyorsabb számítógépe sem biztos, hogy végez a feladattal, amíg kihűl a nap. Hiába építünk egyre gyorsabb számítógépeket, biztos, hogy azok is belevesznek a feladat számtengerébe, hiszen ha 100 helyett 101 városra akarjuk megoldani a feladatot, már ez az egy új adat is milliárdokkal növelheti az elvégzendő számítások mennyiségét. Ezt a jelenséget nevezik kombinatorikus robbanásnak.

Sajnálatos módon a számítástechnika legkülönbözőbb területein ütközünk ebbe az osztályba, a Nem Polinomiális (NP) idő alatt megoldható problémák osztályába (azaz a számítások elvégzéséhez szükséges idő a bemenő adatok számának nem polinom, hanem valamely más, gyorsabban növő exponenciális vagy faktoriális függvényével arányosan fejezhető ki). Látható tehát, hogy igenis fontos a számítástechnikai módszerek, és különösen az NP-teljes problémák megoldásának hatékonyabbá tételével foglalkozni. Ez olyan pont, ahol a mesterséges intelligencia kutatások által létrehozott eszközök komoly segítséget adhatnak. Ilyenek például a heurisztikus keresési stratégiák. Heurisztikus stratégián olyan technikákat értünk, amelyek fokozzák a különböző keresések hatékonyságát.

Ha valaki megpróbált eljátszani az első feladattal, akkor számára nyilvánvaló, hogy eközben egy keresést végzett az alkalmazandó műveletsorok között. Azt hiszem, az is sokakkal előfordult, hogy miután találtak egy megoldást, tovább olvasták a szöveget, és nem vizsgálták végig az összes lehetséges megoldást. Megelégedtek azzal, hogy ránézésre elfogadható hosszúságúnak tűnt az általuk talált megoldás. Ez az emberi problémamegoldás egy igen érdekes vonása, amit nyilvánvalóan az evolúció alakított ilyennévé.

Képzeljünk el egy macskát, amint gondosan mérlegel minden faágat, mielőtt valamelyikre felmászik az őt üldöző kutya elől. Nem lenne túl hosszú életű. Itt nyilvánvalóan kompromisszumról van szó. Amíg a macska futtában odaér a fához, ki kell választania egy olyan ágat, amelyen biztonságban van. Nincs szüksége optimális megoldásra, hiszen nem fontos, hogy azon faágak közül, amelyek még kibírják a súlyát, a legmagasabban levőt találja meg. Számára elég, ha a kiválasztott ágat a kutya már éppen nem éri el.

Ez a megközelítés azokra a helyzetekre is igaz, melyekkel az emberek általában szembekerülnek. Többnyire nincs szükség az optimális megoldásra, csak egy elfogadhatóra. Erre példa a parkolóhely-keresés. Az áruház parkolójában az érkező általában beáll az első elég közelinek tűnő helyre, pedig valószínű, hogy van még szabad hely a bejárathoz közelebb is. A korábban említett utazó üzletember példában ilyen stratégia, hogy az utazó mindig a legközelebbi olyan városba utazik, ahol még nem járt. Belátható, hogy ez a kereséshez szükséges időt N! helyett N*N-re csökkenti, ami jelentős javulás, bár az is igaz, hogy ezzel az átlagos úthossz valamivel megnőtt. Az ilyen meggondolásokat nevezik heurisztikának. Ezek az esetek nagy részében az elfogadhatóság határán belül levő megoldásokat szolgáltató stratégiák, amelyek azonban a legrosszabb esetben a matematikai optimumtól nagyon távol eső eredményt is adhatnak.

A mesterséges intelligencia kutatások történetének leglényegesebb eseményei

Először Alan Turing (1912—1954) vetette fel azt a gondolatot, hogy olyan programok konstruálhatóak, amelyek lehetővé teszik, hogy a számítógépek intelligensen viselkedjenek (lásd Turing, A. M.: Computing Machinery and Intelligence; a Computer and Thought c. könyvben).

Maga a „mesterséges intelligencia" kifejezés először 1956-ban Darthmouthban hangzott el, valószínűleg John McCarthy szájából, azon a témában első konferencián, amelyet ő és Marvin Minsky rendeztek. (A résztvevők úgy emlékeznek, hogy tőle hallották először, ő maga viszont azt mondja, hogy már ő is valaki mástól hallotta.) Ez után a konferencia után hozták létre az első kutatócsoportokat a mesterséges intelligencia kutatásoknak a későbbiekben talán legmeghatározóbb egyéniségei: John McCarthy — Stanford Egyetem, Alan Newell és Herbert Simon — Carnegie-Mellon Egyetem (akkoriban még Carnegie Institute of Technology), Marvin Minsky — Massachusettsi Műszaki Egyetem (MIT).

Ezután viszonylag csendes munka folyt 1981-ig, a Tokióban rendezett Ötödik Generációs Számítógéprendszerek konferenciáig, ahol a japán rendezők olyan nagyméretű fejlesztési program beindítását jelentették be, amely a mesterséges intelligencia kutatások eredményeire támaszkodik. Ezzel egy csapásra reflektorfénybe hozták ezeket az eddig viszonylag szűk körben számon tartott kutatásokat. Nemzetközi összefogást javasoltak, ami végül is nem jött létre, viszont a konferencián elhangzottak hatására számos kutatási program indult.

Tulajdonképpen ekkor vette észre a számítástechnikai világ, hogy időközben lehetségessé vált a mesterséges intelligencia kutatások eredményeinek gyakorlati hasznosítása. Ez elsősorban az integrált áramköri technológia fejlődésének köszönhető, ami a mikroprocesszorok és a RAM-áramkörök árának rendkívüli csökkenését okozta, lehetővé téve olyan számítástechnikai módszerek gazdaságos alkalmazását, amelyekről korábban szó sem lehetett nagy számítástechnikai teljesítményigényük miatt. Tipikusan ide tartozik a mesterséges intelligencia kutatások legtöbb technikája, amelyeknek így már gyakorlati alkalmazásuk is kifizetődő.

SOMOGYVÁRI KÁROLY

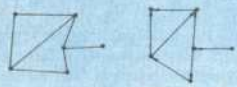
Ajánlott irodalom

— *Dewdeney, A. K.: Számítógépes észjáték. Tudomány, Számítógép-Szoftver különszám*
— *Manna Zohar: Programozáselmélet. Bp., 1981. Műszaki Könyvkiadó*
— *Rich Elaine: Artifical Intelligence. McGraw-Hill, 1983.*
— *Pearl Judea: Heuristics — Intelligent Search Strategies for Computer Problem Solving. Addison—Wesley, 1984.*
— *Minsky, M.—Papert, S.: Perceptrons. MIT Press, Cambridge Mass., 1972.*
— *Garey, Michael R.—Johnson, David S.: Computers and Intractability — A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979. San Francisco.*

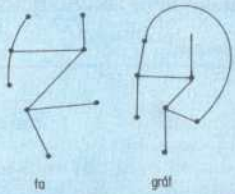
FOGALMAK

problématér: A problémát leíró rendszer lehetséges állapotainak halmaza.

gráf: Olyan matematikai objektum, amit az őt alkotó pontok és élek halmaza határoz meg. Az élek lehetnek irányítottak, ekkor irányított gráfról beszélünk.



fa: Olyan gráf, amelyben nincsenek körutak.



produkciós rendszer: A formális problémamegoldás alapvető eleme, amely a megoldás keresését a rendszer lehetséges állapotainak terében történő bolyongásként írja le.

keresési stratégia: A problématérbeli bolyongást irányító elv.

heurisztika: Olyan, matematikailag gyakran csak körülményesen igazolható, bár az ember számára nyilvánvalónak tűnő technika, amely általában a teljesség irányában tett kismértékű engedményért cserébe az eredmény jóval gyorsabb, hatékonyabb elérését teszi lehetővé.

NP-teljes feladatok: A kombinatorikus feladatok (keresések, rendezések) egy olyan osztálya, amelybe tartozó feladatok megoldásához szükséges idő a bemenő adatok polinomfüggvénynél gyorsabban növő függvényével fejezhető ki.

BASIC és gépi kód

Legutóbb két különleges indexelt címzési módról volt szó. Most két utasításcsoporttal ismerkedünk meg. Az ezekhez tartozó utasítások közös jellemzője, hogy végrehajtásuk során csak az állapotregiszter bitjei változnak, minden más változatlan marad.

Az összehasonlító utasítások

Három utasítás tartozik ebbe a csoportba: a CMP, a CPX és a CPY. Nevük az angol „compare" (összehasonlít) szóból származik. A CMP utasítás az A regiszter tartalmát, a CPX az X-ét, a CPY pedig az Y-ét hasonlítja össze az utasítás operandusa által meghatározott memóriabájt tartalmával, és az eredménytől függően állítják be a Z, C és N feltételbitek értékét.

A Z bit értéke akkor lesz 1, ha a regiszter és a memóriabájt tartalma megegyezik, különben Ø. Ezt kihasználhatjuk az összehasonlító utasítást követő BEQ, illetve BNE utasítással.

A C bit értéke akkor lesz Ø, ha a regiszter tartalma kisebb a memóriabájténál, és akkor lesz 1, ha nagyobb vagy egyenlő. A C bit beállítását többek között a BCC, illetve a BCS ugróutasítások használatával tudjuk kihasználni.

Az N bit értéke akkor lesz 1, ha a regiszter tartalma kisebb a memóriabájténál, és akkor lesz Ø, ha nagyobb vagy egyenlő. Az összehasonlítás eredményét a BPL és BMI feltételes ugróutasításokkal használhatjuk fel.

Úgy látszik, mintha az összehasonlító utasítások végrehajtása után a C és N feltételbitek állása éppen ellenkező lenne, s így a BCS és BPL, illetve BCC és BMI feltételes ugróutasításokat bármikor felcserélhetnénk egymással. Vigyázat, nem így van!

A C bit beállítása szempontjából az összehasonlítás előjel nélküli értékekre történik, viszont az N bit beállításánál az összehasonlítandó bájtokat előjeles számoknak tekinti a processzor. Erről ne feledkezzünk meg!

Mostani egyetlen példaprogramunk ugyanazt a feladatot oldja meg, mint az előző hónapban közölt második példaprogram, de a betűk megjelenítése növekvő sorrendben halad. Ehhez egy CPY utasítást be kell iktatni annak vizsgálatára, hogy elértük-e azt az Y értéket, amelynél a ciklust be kell fejezni. A $C00D címen lévő BCC utasítás helyett itt megfelelne BMI vagy BNE utasítás is.

Az egyes feltételbitek értékét állító utasítások

Ebbe a csoportba hét utasítás tartozik, melyeket két alcsoportba sorolhatunk. A SEC, SED és SEI a nevük utolsó betűje által meghatározott feltételbitek értékét 1-re állítják (SEt), míg a CLC, CLD, CLI és CLV ugyanezen bitek értékét Ø-ra állítják, vagyis törlik (Clear).

A SEC és CLC utasításokkal gyakran fogunk találkozni, alkalmazásuk széles körű. A következő alkalommal az aritmetikai utasítások példáján lesz róluk szó.

A SEI és CLI utasításokat megelőzően szólnom kell az I feltételbittel kapcsolatos tudnivalókról.

A 65xx családhoz tartozó mikroprocesszorral működő számítógépek ROM-jában a $FFFE . . . $FFFF című bájtokon egy memóriacím található, az úgynevezett maszkolható megszakítási kérelmeket feldolgozó rutin címe. Ezt a megszakítási kérelem angol nevének (Interrupt ReQuest) rövidítése alapján IRQ-rutinnak nevezik. A megszakítási kérelmeket általában hardver események — legismertebb ezek közül az órajel-generátor által 1/60 másodpercenként kiadott IRQ — idézik elő.

A maszkolható megszakítási kérelem jelentkezésekor a processzor befejezi az éppen végrehajtott utasítást, majd megvizsgálja az I feltételbit állását: ha ez Ø, akkor az IRQ rutinra adja a vezérlést, úgy, mintha egy JMP ($FFFE) utasítást hajtana végre. Az IRQ rutin végrehajtása után az eredeti program folytatódik azzal az utasítással, amely a megszakítási kérelem végrehajtásakor következett volna. Ha a bit értéke 1, a processzor figyelmen kívül hagyja a megszakítási kérést, és az eredeti program futása azonnal folytatódik.

A megszakításkezelésről most csak annyit kell tudni, hogy a SEI utasítással megtilthatjuk a processzornak a maszkolható megszakítási kérelmek végrehajtását, és a CLI-vel újra megengedhetjük azt. Egyelőre ne kísérletezzünk ezeknek az utasításoknak a használatával. Rövidesen gyakorlati példákon mutatom be alkalmazásukat.

A D bit — mint azt már korábban említettem — a gépi kódú decimális aritmetika kapcsolója. Bekapcsolt állapota a decimális, törölt állapota pedig a bináris aritmetika használatát jelzi a processzornak. Mivel a Commodore gépek beépített programjaiban nincs decimális aritmetika, ezért a jelen sorozatban megismert gépek a bekapcsolás után negyedikként egy CLD utasítást hajtanak végre.

```
c000   a901     lda #$01
c002   a000     ldy #$00
c004   990004   sta $0400,y
c007   9900d8   sta $d800,y
c00a   c8       iny
c00b   c028     cpy #$28
c00d   90f5     bcc $c004
c00f   60       rts
```

Ne tévesszük össze a gépi kódú decimális aritmetikát a BASIC-ben használatossal. Míg az előbbi a processzornak egy üzemmódját jelenti, az utóbbit az interpreter konverziós rutinjai valósítják meg, többnyire nagyon bonyolultan.

A decimális aritmetika használatával részletesebben majd később ismerkedünk meg, de már most felhívom a figyelmet arra, hogy a SED utasítás végrehajtását megelőzően egy SEI-vel le kell tiltani a megszakítást, és csak a CLD után szabad CLI-vel újra engedélyezni. Ez azért fontos, mert a megszakítási rutin — a többi beépített rutinnal megegyezően — bináris aritmetikára van felépítve, és a D bit bekapcsolt állásánál minden összekeveredik.

A CLV utasításnak nincs párja, ugyanis a V bitet szoftver úton közvetlenül csak törölni tudjuk. Az ezzel kapcsolatos tudnivalókról is később lesz szó.

BARNA LÁSZLÓ

3 éve működő
EC 5567, EC 5067.02, EC 5667
bolgár gyártmányú 800 Mbyte
kapacitású mágneslemezsor
eladó.
A gépek működés közben
megtekinthetőek.
Cím:
SZENZOR
SZERVEZÉSI VÁLLALAT
Számítóközpont
Budapest XIII., Lehel út 11.
Telefon: 202-429, 401-539

Teknősbéka-grafika BASIC-ben

A LOGO nyelvet a gyerekek számára különösen az teszi nagyon egyszerűvé, hogy van egy olyan eszköze, amely valósággal odaláncolja a számítógéphez az embert, újabb és újabb öntanító, felfedező kísérletezésre serkentve. Ez az eszköz a teknősbéka-grafika.

Sajnos a BASIC-ben nincs meg ez, és — mert a LOGO fordítóprogramok sokkal nagyobb tárterületet használnak, mint a BASIC fordítók — kis tárkapacitású gépeken meg sem valósítható. Ismertek olyan fordítók, amelyek a LOGO nyelvnek csak a teknősbéka-grafikára szűkített részét tartalmazzák. Ezek a torzók azonban nyelvként túl keveset nyújtanak, és a tárigényük még így is sokszor nagyobb, mint a BASIC fordítóké.

Talán ki lehetne egészíteni a BASIC nyelvet a teknősbéka-grafikával? Ez azonban a két nyelv szerkezetének teljesen különböző volta miatt nem vezet eredményre. Végül van még egy lehetőség: olyan BASIC programot kell írni, amelyik megvalósítja a teknősbéka-grafikát, ki lehet lépni belőle, vissza is lehet térni, miközben bizonyos lényeges információk megmaradnak változatlanul (így pl. mód van egy — a legtöbb LOGO-változatból hiányzó — utasítás, a képernyőtartalom-nyomtatás megvalósítására is).

Olvasóinknak, ha nem jártasak a LOGO nyelvben, az OTTHON ÉS TECHNIKA c. lapban az elmúlt évben indult sorozatot ajánlom. Ebben lépésről lépésre (és kis lépésekkel, sok magyarázattal, példával) halad az ismertetése: mindenről szó esik, ami számunkra szükséges.

A BASIC programmal kapcsolatosan az ÖTLET 1983. október 20-i számának 31. oldalán szerepel egy feltételes ígéret a legtöbb gépen futtatható változat nyilvánosságra hozatalára. Az 1984. március 1-jei szám 17. oldalától közöltek is egy ZX—Spectrum gépre készült változatot, mely a szerző szerint a LOGO-parancsok közül a felsoroltakat — FD, BK, RT, LT, REPEAT, PU, PD, HOME, CLEARSCREEN — ismeri. (A REPEAT-ciklus kezdetét külön nem jelzi, a végét pedig szögletes zárójel helyett egy külön kulcsszóval.) Tud eljárást definiálni és meghívni. Ismeri a rekurziót is. Nem ismeri az IF . . . THEN szerkezetet. (A számunkra közömbös hiányokat nem említem.) A program kb. 220 sor volt, aminek mintegy tizede megjegyzés. Más gépen futtatható változatot nem ismertettek, és az átíráshoz tanácsokat nem adtak.

A mi sorozatunk először egy csak keveset tudó, egyszerű megoldást, majd a továbbiakban már az egyre többre képes változatokat ismerteti. A főcél a programozás, és csak mellékesen számít, hogy a program mennyire fejlett. Célunk az is, hogy információkat nyújtsunk azok számára, akik át szeretnék írni a programot saját gépükre. Szívesen adunk ehhez tanácsokat, és természetesen közöljük az érdekes tapasztalatokat.

Rajzoló 1.

Hogyan kezdjünk hozzá? Minden kicsit is bonyolultabb esetben először meg kell fogalmazni a feladatot. Első kísérletre a megfogalmazás csak ritkán lesz végleges, mert menet közben (gyakran csak a már késznek hitt program futtatása után) derül ki, hogy módosítanunk kell. Így — bár törekedjünk rá, hogy a feladat értelmezése és megoldása szabatos, teljes körű, egyértelműen meghatározott legyen, — ne essünk kétségbe és ne adjuk fel, ha tapasztaljuk, hogy erre és arra az esetre nem gondoltunk, ebben és abban az esetben mást kellene a programmal csináltatnunk, hanem javítsuk ki a hibákat.

Csak a feladat megfogalmazása után készítsük el a megoldás menetét ismertető leírást. Ez lehet folyamatábra, tényleges (szöveges) leírás stb. Kísérelijük meg számítógép nélkül kipróbálni, hogy a leírás megfelel-e a feladatnak, vagyis a várt eredményt adja-e valamilyen ellenőrizhető esetben. Ezután gyakran egy, a leírást finomító lépés következik. Általában ez azért hasznos, mert a leírás egyes lépései túlságosan összetettek ahhoz, hogy a nekik megfelelő programrészletet könnyen meg lehessen írni.

Az eddigieket követi a program megírása és ellenőrzése a leírással azonos módon, kiegészítve próbafuttatással. Rendszerint nem kerülhető el az átdolgozás, amelynek során gyakran többször is vissza kell térnünk az első lépéshez.

E látszólag hosszú, de valószínűleg nem haszontalan bevezető után térjünk rá a konkrét eset tárgyalására. Első változatunk meglehetősen szimpla lesz. Ezt az *1. ábrán* látható bejelentkezése bizonyítja. A szövegben a telexüzeneteknél szokásos átírás szerint:

Á = AA
É = EE
Ó = OO
Ö = OE
Ú = UU
Ü = UE

hosszú ő, ű pedig nincs. (Ezt az átírást azért választottam, mert minden gépen megvalósítható.) A parancsok elválasztójele eltér a LOGO nyelvétől: pontosvessző a " " helyett. Ennek az az oka, hogy sokan szinte automatikusan kihagynak a kulcsszavak után üres helyet, míg mások nem. Úgy döntöttem hát, hogy üres helyet nem használok (sőt, használatát nem is engedélyezem), hanem egy másfajta elválasztójelet vezetek be.

Írjuk össze a feladatokat:

1. Alapállapotba kell állítanunk a programot.
2. A programnak várni kell a parancsainkra.
3. Ha nem kapta meg, vissza a 2. ponthoz, ha igen, ellenőrizze, hogy „értelmes" parancsokat kapott-e.
4. Ha nem, akkor (és ez eltérés a LOGO nyelvtől), vissza a 2. ponthoz (a LOGO nyelvben hibaüzenet után tér vissza), ha igen, dolgozza fel.
5. Állapítsa meg, hogy melyik a következő parancs, és annak megfelelően határozza meg a teknős új helyét vagy új irányát, illetve, ha már nincs több parancs, vissza a 2. ponthoz.
6. Új helymeghatározás, attól függően, hogy „előre" vagy „hátra" parancsot kapott, majd rajzolás.
7. Új iránymeghatározás attól függően, hogy „jobbra" vagy „balra" parancsot kapott, majd vissza az 5. ponthoz.
8. Rajzolja fel a teknős új helyét, majd vissza az 5. ponthoz.

Ha ezek alapján elkészítjük programunkat, kellemetlen meglepetések érhetnek. Így pl. nekünk kell kiszámítanunk a teknős minden új helyzetét, mielőtt odairányíta-

nánk, nehogy a rajzmezőn kívüli érték adódjon, mert ha mégis, akkor a program hibaüzenettel leáll. Be kell tehát építenünk egy ezt megakadályozó ellenőrzést. Például: a rajzterületen kívüli értéket hagyja figyelmen kívül, a rajzterületen kívüli érték helyett a határnak megfelelőt írja be (ezt választottam), a rajzterületet elhagyva a teknős jöjjön vissza az ellenkező oldalon (sok LOGO-változat is ilyen).

A kiegészítésen kívül más problémák is adódnak. Az 5—8. pontban kijelölt feladatok túl bonyolultak ahhoz, hogy egy kezdő vagy középhaladó programozó az ennek megfelelő programrészt fel tudja írni. Célszerű tehát a finomítás.

Hogyan tudjuk a teknős helyét és irányát meghatározni? Ismernünk kell a kezdeti értékeket (ez a rajzmező geometriai középpontja — derékszögű koordinátaértékekkel megadva — és a rajzmező felső élének közepe felé mutató irány), valamint minden parancs után a változást. Ez utóbbihoz ismernünk kell az elmozdulás vagy az elfordulás nagyságát. Az elmozdulás nagysága megegyezik az FD ill. BK parancsot követő szám értékével. Az irányváltozás nagysága pedig az RT, ill. az LT parancsot követő szám. Egyszerre nem változhat a hely és az irány, csak az egyik, ezért a parancsokat követő számértéket minden esetben D-vel jelölhettük. A teknős helyét, X, Y irányát az FI változó adja meg. Az új X, Y értéket a következő egyenletekből számíthatjuk:

$$XUJ = XRE + (D * COS(FI))$$

$$YUJ = YRE - (D * SIN(FI))$$

(Y negatív irányítása azoknál a gépeknél helyes, ahol az origó a rajzmező bal felső sarka.) „Hátra" parancs esetén D értéke negatív. Az új irány pedig:

$$FIUJ = FIRE + (D * PI/180)$$

(A PI — az ismert körkerület/átmérő hányadosnak a gépünk által megadott pontosságú közelítése; ha BASIC fordítóprogramunkban a PI mint függvény nincs beépítve, 8 tizedesjegy pontossággal előállítható a

$$355/113$$

osztással). A szokásoknak megfelelően a balra forgatást vettem kiinduló értéknek, ezért a jobbra forgást számolni kell a következő egyenlettel:

$$DBA = 360 - DJO$$

Célszerű a szögértékeket 0—360 fok között tartani, ezért az ezen kívülre eső értékeket módosítanunk kell. Ezzel nem okozunk hibát, mert mindkét szereplő szögfüggvény e szerint periodikus. Tehát, ha a szög nagyobb lesz, mint 360 fok, le kell vonnunk belőle 360-at.

Nem tartozik általában a program készítőjének alapfeladatához, de célszerű a program későbbi felhasználóját a munka kezdetén tájékoztatnunk arról, hogy mit nyújt a program és hogyan dolgozzék vele. Írjuk majd hát ezt is bele programunkba mint 0. pontot.

Most már elkezdhetjük a programozást! A program ugyan DRAGON BASIC-ben íródott, de az átíráshoz szükséges információkat közlöm. A 20., 30. sor a bemutatás (0. pont). A 40. sorban beállítjuk a kezdeti feltételeket (1. pont). Itt az első utasítás egy helyfoglalás a karaktersorozatoknak, a legtöbb BASIC-változatnál elhagyandó, X0, Y0, ALFA a kezdeti értékek (ennél a BASIC-változatnál a változónevek 8 karakter hosszúak lehetnek, ahol ez nem megengedett, kurtítsuk a nevet), a PMODE, PCLS a grafikus üzemmód beállítása és a képmező törlése (ez a legtöbb gépnél egyedi megoldású; tanácsot szívesen adok az átíráshoz). Az utolsó utasítás a képernyő X és Y koordinátákkal megadott pontját köti össze a teknős eddigi helyével (ez egyes gépeken, amelyeken egyenest rajzoló utasítás van, azzal helyettesítendő, amelyiken nincs, ott ilyen szubrutint kell írni — ebben is tudok segíteni). Az 50. sor megvizsgálja, hogy volt-e beadás, ha igen, az A1$-ben tárolja (az én nyomtatóm a svéd ABC-vel ír, ezért a $ helyett mást nyomtatott), a karakterek számát pedig A-ban (2. pont). Néhány gépnél a karaktersorozatokkal kapcsolatos függvények másak, mint az 50., 80., 90., 100. és 120. sorbeliek. Ezekre kérés esetén külön kitérek.

A 60. sorban elkezdjük A1$ feldolgozását (4. pont).

A 70. sorban nézzük, hogy van-e még karakter, egyúttal növeljük egy segédváltozó értékét (5. pont).

A 80. sorban megvizsgáljuk, hogy a következő karakter elválasztójel-e, ha nem, folytatjuk a vizsgálatot a következővel (5. pont).

A 90. sorban leválasztjuk A1$-ból a megtalált parancsot A$-t (5. pont).

A 100. sorban ebből leválasztjuk az elválasztójelet, de ha már a parancssorozat végén vagyunk, akkor nem (5. pont).

A 110. sorban eldöntjük, hogy a szög nagyobb-e, mint 360 fok, és ha igen, ennyivel csökkentjük az értékét (7. pont).

A 120. sorban a parancsot kulcsszóra (C$) és egy számra (D) választjuk szét. A negyedik utasítás bekapcsolja a grafikus képernyőt. (Ez is más és más lehet gépenként). Az utolsó utasítással az elmozdulást okozó parancsokat feldolgozó részre ugrunk, ha ilyet adtunk be (5. pont).

A 130. és 140. sorban szétválasztjuk a két irányt változtató parancs feldolgozását (7. pont).

A 150. sorban az értelmetlen parancsot kiküszöböljük. A elejére be lehet írni egy hibaüzenetet is, pl.

PRINTA$, "-T TALAALTAM. EZT NEM ISMEREM!";

A 160. sorban balra forgó szögre váltunk át (7. pont).

A 170. sorban meghatározzuk az új irányt (7. pont).

A 180. sorban ugrunk a további feldolgozó részre (7. pont).

A 190. sorban „előre"-menetre váltunk át (6. pont).

A 200. sorban meghatározzuk az új helyet (6. pont).

A 210. sorban rajzolunk, ha a rajzmezőn belül vagyunk (8. pont).

A DRAW utasítás sok gépen létezik, de különböző dolgokat jelent. Itt a RAJZ$ karaktersorozatban megadott tennivalókat hajtja végre. A lehetséges tennivalók közül itt most csak a 40. sornál említett szerepel.

A 220—250. sorban a rajzmezőn belül tartjuk a teknőst (6. pont).

Végül a 260. sorban folytatjuk a feldolgozást, majd amikor már nincs mit, várjuk az új parancsokat (5. pont).

Dr. Simonyi Endre

```
10 REM RAJZOLO I
20 PRINT"PARANCSOK:":PRINT"FD-ELOERE":PR
INT"BK-HAATRA":PRINT"LT-BALRA":PRINT"RT-
JOBBRA":PRINT"MINDEGYIK UTAAN EGY SZAAM
AALL.":PRINT"A PARANCSOK SOROZATBA KAPCS
OLHATOOK."
30 PRINT"AZ ELVAALASZTOJEL A PONTOSVESSZ
OE."
40 CLEAR2000:X0=128:Y0=96:FI=355/113:ALF
A=PI/2:FI=ALFA:X=X0:Y=Y0:PMODE4,1:PCLS:R
AJZ$="M ";X;",";Y;"
50 B$=INKEY$:IFB$="" THEN50 ELSE IN"UTA1
$:A=LEN(A1$)
60 FOR I=1TO A:J=0
70 J=J+1:IF J=A-I+1 THEN90
80 J$=MID$(A1$,I-1+J,1):IF J$<>";" THEN7
0
90 A$=MID$(A1$,I,J):I=I+J-1
100 IF J$=";" THEN A$=LEFT$(A$,LEN(A$)-1
)
110 IF FI>2*PI THEN FI=FI-(2*PI)
120 C$=LEFT$(A$,2):D$=MID$(A$,3,LEN(A$)-
2):D=VAL(D$):SCREEN1,1:IF (C$="FD")OR(C$
="BK") THEN190
130 IF C$="LT" THEN170
140 IF C$="RT" THEN160
150 GOTO260
160 D=360-D
170 FI=FI+(D*PI/180)
180 GOTO260
190 IF C$="BK" THEN D=-D
200 X=X+(D*COS(FI)):Y=Y-(D*SIN(FI))
210 IF(X>-1)AND(X<256)AND(Y>-1)AND(Y<192
) THEN DRAW RAJZ$:GOTO260
220 IFX<0THEN X=0
230 IFX>255THEN X=255
240 IFY<0THEN Y=0
250 IFY>191THEN Y=191
260 NEXTI:GOTO50
```

MERGE

Az OVERLAY, CHAIN, APPEND eljárásoknál mindig egy, a tárban elhelyezkedő programhoz kapcsoltunk hozzá további programokat. A hozzákapcsolt programot vagy az eredeti program helyére, vagy annak végére töltöttük be. A múlt havi cikkben egy olyan különleges eljárást mutattam be, amikor a hívó program tetszőleges hosszban a tárban maradhat, és csak az egy adott sorszámnál nagyobb sorszámú programsorok helyére töltődik be az új program.

A most vizsgált MERGE eljárással alapjában más módszert követünk. Most nem az a cél, hogy egy futó programhoz kapcsoljunk újabb programrészeket, hanem az, hogy több külön-külön is működő programból ollózzunk össze egy újabb működőképes programot. Ezzel az eljárással tehát egy tetszés szerinti programból kivágunk egy számunkra hasznos programrészt, majd ezt beolvasztjuk az éppen most készülő programunkba. Így nemcsak a program végéhez kapcsolhatunk új részeket, hanem a program belsejébe is saját részeket szúrhatunk be. A módszer hasonlít ahhoz, amikor a BASIC szerkesztő segítségével a már meglevő programsorok közé illesztünk be néhány új sort. Ilyenkor az új BASIC-sor mindig a sorszámának megfelelő helyre íródik be. Ha volt már ilyen sorszámú sor a programban, akkor ez az új sorra cserélődik ki.

Az itt közölt módszer előnye, hogy több sorból álló programrészeket is beollózhatunk a programunkba, elkerülve a hosszadalmas begépelést. Az eljárásban most három programmal dolgozunk:

— az első az a program, amelybe be akarunk építeni egy új programrészt;

— a második a beépítendő programrész;

— a harmadik az előző két program egyesítéséből keletkező új program.

A feladat elvégzéséhez először előkészítjük az első programot. Ez annyiból áll, hogy a programot úgy sorszámozzuk, hogy fogadni tudja a beépítendő programrészt. Az így előkészített programot lemezen tároljuk.

Ezután a második programból eltávolítjuk a fölösleges sorszámú utasításokat, majd a megmaradt részt úgy sorszámozzuk át, hogy az első program megfelelő helyére be tudjuk majd vinni. Az így kapott programot szintén lemezen tároljuk.

Most bevisszük a MERGE programot, és elindítjuk. A program kérdéseire megadjuk az előbb előkészített két program nevét, majd az így készülő program nevét is. Ezután a MERGE program felváltva olvassa az első, illetve a második program sorait. Az új programba egészen addig az első program sorai kerülnek, amíg az első program sorszámai kisebbek a második program sorszámánál. Két azonos sorszámnál mindig a második programból visszük át az éppen aktuális programsort.

A második program sorait egészen addig folyamatosan másoljuk át a készülő új programba, míg olyan sorszámhoz nem jutunk, amelynél ismét van kisebb sorszámú sor az első programban. Ekkor ismét az első program sorait kezdjük átmásolni.

A MERGE program tehát ugyanazt végzi el automatikusan, amit programszerkesztéskor a kézi beírással nekünk kellene megcsinálnunk.

A program ismertetése

A felhasznált fontosabb változók:

N1$, N2$, N3$	a három program neve;
P (1—3)	az éppen feldolgozás alatt levő program száma;
FI (3—5)	a három programhoz rendelt fájl száma;
S3$, S4$	a 3., illetve a 4. fájlból kiolvasott programsor;
S3, S4	a 3., illetve a 4. fájlból kiolvasott programsor sorszáma;
H1	lemezhiba-vizsgálat eredménye; Ø=OK;

```
10000  REM   MERGE PROGRAM FILES
10010  REM ----------------------
10020  REM
10030  PRINT "{CLR}{CD}{CD}": SPC( 8);
       "{RVS} MERGE PROGRAM FILES {RVO}
       "
10040  PRINT  SPC( 8);"{RVS}
                {RVO}"
10050  PRINT  SPC( 8);"{RVS} PROGRAM
       OSSZEVAGAS {RVO}": PRINT :
        PRINT
10060  INPUT "1.FORRAS PROGRAM NEVE:";
       N1$
10070  INPUT "2.FORRAS PROGRAM NEVE:";
       N2$
10080  INPUT " KESZULO PROGRAM NEVE:";
       N3$
10090  OPEN 15,8,15
10100  S3$ = "": S4$ = ""
10110  P = 1: OPEN 3,8,3,N1$: GOSUB 60
       030: IF A THEN  CLOSE 3: GOTO 10
       110
10120  IF H1 THEN 10310
10130  P = 2: OPEN 4,8,4,N2$: GOSUB 60
       030: IF A THEN  CLOSE 4: GOTO 10
       130
10140  IF H1 THEN 10310
10150  P = 3: OPEN 5,8,5,N3$ + ",P,W":
        GOSUB 60300: IF A THEN  CLOSE 5:
        GOTO 10150
10160  IF H1 THEN 10310
10170  PRINT "{HOME}{CD}{CD}{CD}{CD}
       {CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}"
10180  FI = 3:S$ = ""
10190  GOSUB 30030: GOSUB 30030:
        PRINT# 5,S$;
10200  FI = 4: GOSUB 30030: GOSUB 3003
       0
10210  GOSUB 40030
10220  GOSUB 50030
10230  IF S3 <  = S4 THEN 10260
10240  IF  LEN(S4$) > 2 THEN  PRINT "
       {CU}";: GOSUB 60180: PRINT "{CU} PR
       O 2";S4,",SOR MASOLASA  ":
        PRINT# 5,S4$;
10250  GOTO 10220
10260  IF  LEN(S3$) > 2 THEN  PRINT "
       {CU}";: GOSUB 60180: PRINT "{CU} PR
       O 1";S3,",SOR MASOLASA  ":
        PRINT# 5,S3$;
10270  IF S3 = S4 AND S3 < 65536 THEN
       10210
10280  GOSUB 40030
10290  IF S3 <  > 65536 OR S4 <  > 655
       36 THEN 10230
10300  PRINT# 5, CHR$(0); CHR$(0);
10310  GOSUB 60130: GOSUB 60200:
        PRINT : GOSUB 60060: IF A THEN 10000
10320  END
```

```
60000  REM#DISC HIBA VIZSGALAT
60010  REM ----------------------
60020  '
60030  A = 0: INPUT# 15,H1,H$,H2,H3
        IF H1 < 20 THEN H1 = 0: RETURN
60040  '
60050  GOSUB 60130: PRINT "PRG";P;"
       {RVS}DISC HIBA{RVO}"H1;H$;H2;H3
60060  PRINT "{CD}      {RVS}F1{RVO} V
       EGE    {RVS}F7{RVO} ISMETLES"
60070  GET A#: IF A# =  CHR#(136)
        THEN A = 1: GOTO 60090
60080  IF A# <  > CHR#(133) THEN 6007
       0
60090  GOSUB 60130: GOSUB 60180:
        GOSUB 60180: GOSUB 60180: RETURN
60100  REM   HIBA KIIRAS HELYE
60110  REM ----------------------
60120  '
60130  PRINT "{HOME}{CD}{CD}{CD}{CD}
       {CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}
       {CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}";
        RETURN
60140  '
60150  REM  KEPERNYO SOR TORLES
60160  REM ----------------------
60170  '
60180  PRINT "
                    ": RETURN
60190  '
60200  REM  HIBA UZENET TORLES
60210  REM ----------------------
60220  '
60230  GOSUB 60130:GOSUB20630: GOSUB 6
       0230: GOSUB 60230: RETURN
60240  '
60250  REM     CLOSE FILES
60260  REM ----------------------
60270  '
60280  CLOSE 3: CLOSE 4: CLOSE 5:
        CLOSE 15: RETURN
60290  '
20000  REM# 1 PROGRAMSOR OLVASAS
20010  REM ----------------------
20020  REM
20030  S$ = ""
20040  GOSUB 30030: GOSUB 30030
20050  IF S1# =  CHR#(0) THEN 20100
20060  GOSUB 30030:S =  ASC(S1$)
20070  GOSUB 30030:S =  ASC(S1$) # 256
        + S
20080  GOSUB 30030
20090  IF S1$ <  > CHR$(0) THEN 20000
20100  RETURN
20110  '
30000  REM   1 KARAKTER OLVASASA
30010  REM ----------------------
30020  REM
30030  GET#FI,S1$
30040  S1$ =  LEFT$(S1$ +  CHR#(0),1)
30050  S$ = S$ + S1$
30060  RETURN
30070  '
40000  REM   1.PROGRAM OLVASASA
40010  REM ----------------------
40020  REM
40030  '
40040  IF  LEN(S3$) = 2 THEN S3 = 6553
       6: GOTO 40060
40050  FI = 3: GOSUB 20030:S3$ = S$:S3
        = S
40060  RETURN
40070  '
50000  REM   2.PROGRAM OLVASASA
50010  REM ----------------------
50020  REM
50030  '
50040  IF  LEN(S4$) = 2 THEN S4 = 6553
       6: GOTO 50060
50050  FI = 4: GOSUB 20030:S4$ = S$:S4
        = S
50060  RETURN
50070  '
```

A válaszkiértékelés: Ø=vége, 1=ismétlés.

A főprogram

10000—10080	kezdő képernyő elkészítése, programnevek megadása;
10090	hibacsatorna megnyitása;
10100—10170	a 3 programnak megfelelő 3 fájl megnyitása;

10180—10190	az 1. program betöltési címének átmásolása a 3. programba;
10200	a 2. program betöltési címének átlépése;
10210	egy programsor olvasása az 1. programból;
10220	egy programsor olvasása a 2. programból;
10230—10290	az átmásolandó sorszám kiválasztása és a kiválasztott programsor átmásolása;
10300—10320	a program végét jelző nullák felírása és a program befejezése.

A szubrutinok

Egy programsor olvasása

A programsorok felépítését részletesen ismertettem a Programtitkosítás c. cikkben a Mikromagazin 1986. 10. számában.

20040—20050	A következő programsor mutatójának beolvasása. Ha a mutató <255 (a második bájt 0), akkor vége.
20060—20070	A sorszám 2. bájtjának beolvasása és elhelyezése az S változóban.
20080—20090	A programsor további részeinek olvasása a program végét jelző 0 bájtig.

Egy karakter olvasása

30030	Az FI által kiválasztott fájlból egy karakter beolvasása az S1$-ba.
30040—30050	Az üres karakter konvertálása CHR$ (0)-ra, és az olvasott karakterek gyűjtése az S$-ban.

A kiválasztott program olvasása

40040	Ha az utolsó sort olvasta, akkor a sorszám beállítása a maximálisra, hogy a másik programot tovább tudjuk olvasni.
40050	A sor olvasása az előző két rutin segítségével.

Lemezhiba-vizsgálat

60030	A hibacsatorna olvasása.
60050	A hiba kijelzése a képernyőn.
60060—60090	Választási lehetőség az ismétlés vagy a befejezés között.

A többi kis szubrutin nem igényel bővebb magyarázatot. A már közölt APPEND és a mostani MERGE eljárások használatához feltétlenül szükséges, hogy az összekapcsolt programrészek sorszámai a készített új programnak megfeleljenek. Ehhez a következő, RENUMBER c. cikkben adok segítséget.

ZSOM BÉLA

Vigyázat!

Tolvaj!

Milyen módon generáljunk 20-as típusú hibát a lemezre?

Az eddigiekben megismerkedtünk az egyes lemezvédési technikák elméleti lehetőségeivel. Az egyik ilyen módszer, hogy a lemez bizonyos részére hibát generálunk, és ezt a programból ellenőrizzük. Ilyen hiba jelentkezik például diszk ID-tartalom cseréje esetén, vagy ha nincs adatblokk és szinkronjel. Továbbá, ha hibás a fej ellenőrzősszege vagy egyáltalán nincs blokkfej. Jelenleg ezzel az utóbbival foglalkozunk részletesebben, illetve bemutatunk a Dos Insideből egy programot, amellyel a lemez bármely blokkján vagy sávján ilyen hibát hozhatunk létre. Természetesen az e lemezterületen lévő adatok elvesznek, ezért gondosan járjunk el: győződjünk meg arról, hogy valóban helyes lemezt tettünk-e a meghajtóba.

A program bekéri az illető sáv (track)- és szektorszámot, amelyen a hibát akarjuk generálni. Ezután megkeresi a kijelölt sávon a megadottnál egygyel előbb lévő szektort, majd a következőt – vagyis a megadottat – elrontja.

A Das Grosse Floppy Buch c. könyvben található memóriatérkép nem elég részletes, ezért szükségesnek tartom a programban használt olyan rendszerváltozók ismertetését, amelyek nem közismertek. (Vö. 1987/2. 15. old.) A $0016 memóriacímtől $001A-ig terjedő tárterületet az FDC (Floppy Disk Controller) használja az aktuális bokkfej megőrzésére.

Kovács P. Attila

COMMODORE 64

```
100 REM 20-AS TIPUSU HIBA - VIC 1541
110 DIM D$(11)
120 PRINT"{CLR} 20-AS HIBA - VIC 1541"
130 PRINT"{LE}{ED} A LEMEZT A MEGHAJTOBA"
140 INPUT"{LF}HZ ILLETO IRHATK ES SZEKTORSZAM"
    ;S,SZ:I=5
150 IF I<1 OR I>35 THEN END
160 NBP=256*PEEK(I*2+1)+PEEK(I*2)
170 IF S<0 OR S>NS THEN END
180 INPUT"{LE}BIZTOSAK HZ ADATOK LEHLAK JJ";A$
190 IF A$<>"I" THEN END
200 OPEN15,8,15
210 PRINT#15,"I0"
220 INPUT#15,EN$,EM$,E1$,E5$
230 IF EN$="00" GOTO 280
240 PRINT"{LE}" EN$","EM$","E1$","E5$
250 CLOSE15
260 END
270 REM A PROGRAM ATIRASA A MEGHAJTO
275 REM MEMORIAJABAN
280 IF S<NS THEN S=S+1: GOTO200
285 REM A SZEKTORSZAM CSOKKENTESE
290 S=S-1
300 JOB=I/6
310 GOSUB 370
320 JOB=128
340 GOSUB 370
350 FOR J=0 TO 11
360 FOR I=0 TO 7
370 NEXT J
380 D$(J)=D$(J)+CHR$(D)
390 NEXT I
400 NEXT J
410 I=0
420 FOR J=0 TO 11
430 PRINT#15,"M-W"CHR$(I)CHR$(5)CHR$(0)D$(J)
440 I=I+8
450 NEXT J
460 REM VEGREHAJTAS
470 PRINT#15,"M-W"CHR$(2)CHR$(0)CHR$(1)CHR$(224)
480 PRINT#15,"M-W"CHR$(4)CHR$(0)
490 GET#15,E$
500 IF E$="" THEN E$=CHR$(0)
510 E=ASC(E$)
520 IF E>127 GOTO 480
530 CLOSE15
540 PRINT"{LE}KESZ"
550 END
560 REM ********************************
570 IKY=0
580 PRINT#15,"M-W"CHR$(0)CHR$(4)
CHR$(7)CHR$(5)CHR$(1)CHR$(5)
590 PRINT#15,"M-W"CHR$(1)CHR$(0)CHR$(1)
CHR$(JOB)
600 IKY=IKY+1
610 PRINT#15,"M-R"CHR$(1)CHR$(0)
620 GET#15,E$
630 IF E$="" THEN E$=CHR$(0)
640 E=ASC(E$)
650 IF IKY>200 GOTO 680
660 IF E>127 GOTO 600
670 IF E=1 THEN RETURN
680 CLOSE15
690 PRINT"{LE}IKYS,HEM SIKERULTI!"
700 END
710 REM 20-AS HIBA
720 DATA 32,10,240,32,86,240,160,20
730 DATA 165,20,201,18,144,12,136,136
740 DATA 201,20,144,6,140,201,31,144
750 DATA 1,136,208,249,197,24,144,6
760 DATA 240,4,169,0,133,20,169,0
770 DATA 69,22,69,23,69,24,69,25
780 DATA 133,26,32,54,245,32,86,240
790 DATA 169,255,141,3,28,173,12,28
800 DATA 41,31,9,192,141,12,28,184
810 DATA 0,189,0,28,24,105,1,1
820 DATA 28,232,224,8,208,243,88,234
830 DATA 32,0,234,169,1,76,165,249

A meghajtó memóriájában futó program fordításistása:

100 ; ********************************
110 ; 20-AS HIBA LEHELKEZTETESE
120 ; ********************************
130 ;
140 *=$0300   ;A PROGRAM A $0300-TOL KEZDODIK
150          ;PUFFEREBEN KERUL
160 ; A BASIC PROGRAMBOL ERRE BEMELLITJUK
170 ; A SZUKSEGES VMLIJUCUKAT (TRACK- ES SZEKTORSZAM)
180 ; A MEGHAJTO MEMORIAJABAN
190 JSR $F510 ;A BEMELLITOJT KENDSZERVMLIJUZUK
200          ;MAJD AZ ILLETO BLOKKFEJET
210 JSR$F556  ;MEGKERESI AZ MEMORIABLOKKOT
220 ;
230 LDY#$14
240 LDA $19   ;SZEKTORSZAM
250 CMP#$12
260 BCC ZUNK
270 DEY
280 DEY
290 CMP#$13
300 BCC ZUNK
310 DEY
320 CMP#$1F
330 BCC ZUNK
340 DEY
350 ZUNK INC $18 ;SHVSZMPH (TRACK)
360 CMP $18
370 BCC MENDEK
380 BEQ MENDEK
390 LDA#$00   ;A SZEKTORSZAM FELULIRASA
400 STA $19
410 ;
415 ;A BLOKKFEJ ELLENORZOOSSZEGENEK A
416 ;KISZAMITASA
417 ;
420 MENDEK LDA#$00
430 EOR $16
```

```
440 EOR $17
450 EOR $18
460 EOR $19
470 STA $1A ;AZ ELLENORZOOSSZEG BEIRASA
480 ;
490 JSR $F934 ;A BLOKKFEJ GCR-KONVERZIOJA
500 JSR $F556 ;A KOVETKEZO BLOKK MEGKERESESE
510 LDA#$FF    ;IRASMOD BEKAPCSOLASA
520 STA $1C03
530 LDA$1C0C
540 AND#$1F
550 ORA#$C0
560 STA $1C0C
570 LDX#$00
580 WRITE LDA $0024,X
590 WAIT1 BVC WAIT1
600 CLV
610 STA $1C01
620 INX
630 CPX#$08
640 BNE WRITE
650 WAIT2 BVC WAIT2
660 ;
670 JSR $FE00 ;OLVASASMOD BEKAPCSOLASA
680 ;
690 LDA#$01
700 JMP $F969
```

E program áttanulmányozása után pedig nézzük meg hogyan lehet az illető sáv minden blokkján ezt a 20-as típusú hibát generálni.

```
100 REM 20-AS TIPUSU HIBA EGESZ SAVRA
103 REM INPUT PARAMETER: SAVSZAM
106 REM ********************************
110 DIM D$(24)
120 PRINT"[CLR]20-AS TIPUSU HIBA -VIC1541
130 PRINT"[DOWN]KEREM A LEMEZT A MEGHAJTOBA!"
140 INPUT"[DOWN]SAVSZAM";T
150 IF T<1 OR T>35 THEN END
160 INPUT"[DOWN]BIZTOS? (I/N)";Q$
170 IF Q$<>"I" THEN END
180 OPEN15,8,15
190 PRINT#15,"I0"
200 INPUT#15,EN$,EM$,ET$,ES$
210 IF EN$="00" GOTO 250
220 PRINT"[DOWN]"EN$", "EM$", "ET$", "ES$
230 CLOSE15
240 END
250 REM AZ ADATOK ATIRASA A MEGHAJTO MEMORIAJABA
260 NS=20+2*(T>17)+(T>24)+(T>30)
270 S=NS
280 JOB=176
290 GOSUB 580
300 FOR I=0 TO23
310 READ D
320 M$=M$+CHR$(D)
330 I$=I$+CHR$(0)
340 NEXT I
350 PRINT#15,"M-W"CHR$(0)CHR$(6)CHR$(24)M$
360 REM VEGREHAJTAS
370 PRINT"[DOWN][RVS]A HIBA FELIRASA[OFF] SAVSZAM ";T
380 JOB=224
390 GOSUB 580
400 PRINT#15,"M-W"CHR$(0)CHR$(6)CHR$(24)I$
410 FOR J=0 TO 24
420 FOR I=0 TO 7
430 READ D
440 D$(J)=D$(J)+CHR$(D)
450 NEXT I
460 NEXT J
470 I=0
480 FOR J=0 TO 24
490 PRINT#15,"M-E"CHR$(0)+CHR$(4)
540 CLOSE15
550 PRINT"KESZ!"
560 END
570 REM ************************************
580 TRY=0
590 PRINT#15,"M-W"CHR$(12)CHR$(0)CHR$(2)CHR$(T)CHR$(S)
600 PRINT#15,"M-W"CHR$(3)CHR$(0)CHR$(1)CHR$(JOB)
610 TRY=TRY+1
620 PRINT#15,"M-R"CHR$(3)CHR$(0)
630 GET#15,E$
640 IF E$="" THEN E$=CHR$(0)
650 E=ASC(E$)
660 IF TRY=500 GOTO 690
670 IF E>127 GOTO 610
680 RETURN
690 CLOSE15
700 PRINT"[RVS]NEM SIKERULT[OFF]"
710 END
715 REM ************************************
720 REM A GEPI KODU PROGRAM
725 REM ************************************
730 DATA 32,163,253,169,85,141,1,28
740 DATA 162,255,160,48,32,201,253,32
750 DATA 0,254,169,1,76,105,249,234
760 REM ************************************
770 DATA 169,0,133,127,166,12,134,81
780 DATA 134,128,166,13,232,134,67,169
790 DATA 1,141,32,6,169,8,141,38
800 DATA 6,169,0,141,40,6,32,0
810 DATA 193,162,0,169,9,157,0,3
820 DATA 232,232,173,40,6,157,0,3
830 DATA 232,165,81,157,0,3,232,169
840 DATA 0,157,0,3,232,157,0,3
850 DATA 232,169,15,157,0,3,232,157
860 DATA 0,3,232,169,0,93,250,2
870 DATA 93,251,2,93,252,2,93,253
880 DATA 2,157,249,2,238,40,6,173
890 DATA 40,6,197,67,208,189,138,72
900 DATA 169,75,141,0,5,162,1,138
910 DATA 157,0,5,232,208,250,169,0
920 DATA 133,48,169,3,133,49,32,48
930 DATA 254,104,168,136,32,229,253,32
940 DATA 245,253,169,5,133,49,32,233
950 DATA 245,133,58,32,143,247,169,35
960 DATA 133,81,169,169,141,0,6,169
970 DATA 5,141,1,6,169,133,141,2
980 DATA 6,169,49,141,3,6,169,76
990 DATA 141,4,6,169,170,141,5,6
1000 DATA 169,252,141,6,6,169,224,133
1010 DATA 3,165,3,48,252,76,148,193
```

A SZOFTVERGYÁRTA NAGYJAI

Az elmúlt nyáron amerikai utamon a szerencsés véletlennek — névazonosságunknak — köszönhetően megismerkedtem a MICROSOFT®-cég egyik vezetőjével. Kérésemre elküldte válaszait az általam megfogalmazott kérdésekre azzal, hogy a közlésre lapunknak kizárólagos jogot ad.

Most röviden bemutatom interjúalanyomat és cégét. A MICROSOFT® 12 év alatt a mikroszámítógépes szoftvergyártás egyik legnagyobb vállalata lett. Berobbanó terméke az első mikroszámítógépre írt BASIC-fordító volt. Ennek továbbfejlesztett változata a ma már gyakorlatilag világszabvánnyá vált — és a cég által szinte minden nagy tömegben előállított mikrogéphez szállított — Microsoft BASIC. Hasonlóan sikeres termékük volt a 8088-típusú mikroprocesszoros rendszerekhez kifejlesztett diszk-operációs rendszer, az MS—DOS®. Ezt használja ma az összes IBM PC/XT/AT és kompatíbilis gép. Ezeken kívül számos ismert termékük van mind az alap-, mind az általános célú alkalmazói szoftverek területén (például Microsoft, Windows, Multiplan, Microsoft Word stb.)

Dr. Simonyi Károly (csak névrokonom) a Budapesti Műszaki Egyetem világhírű professzorának — akit villamosmérnökök generációi az egyetem legjobb előadójának tartottak — a fia. Gimnazistaként 17 éves korában, 1965-ben Ural II típusú számítógépen már elkészítette első fordítóprogramját Budapesten. Később Dániában dolgozott az A/S Regnecentralennél, majd folytatta tanulmányait az Egyesült Államokban: először a Berkeley Egyetemen, majd a stanfordin. Karrierje is ezeken a helyeken kezdődött: a UC Berkeley Computer Center, később a Xerox PARC (Palo Alto Research Center — a Xerox kutatóközpontja Palo Altóban, mely városnak egy külön nevű része a híres stanfordi egyetem) alkalmazottja. Hat éve lett a munkatársa a MICROSOFT Corporation-nek, ahol létrehozta és vezeti az Alkalmazási Szoftverek Csoportját.

16 évesen programozott először, és mivel az Ural II típuson más lehetőség nem volt, oktális (nyolcas számrendszerű) gépi kódban. A program egy 80 × 80 méretű bűvös négyzet készítésére szolgált. Az akkori programkészítés nehézségére, bonyolultságára — és az ifjú Simonyi teljesítményének jellemzésére — egyetlen példa: az első program, amelynek készítésében 1962-ben részt vettem, lényegileg egy másodfokú egyenletnek a megoldása volt háromszáz különböző paraméterértékkel. Ez ma a legolcsóbb mikroszámítógépen BASIC-ben néhány sor, és a futási idő néhány perc. A mi programunk több teleírt lap volt, és több mint 3 hónap kellett ahhoz, hogy egyszer jól lefusson. Abban az időben ugyanis a terem nagyságú, elektroncsöves gépek erősen melegedtek, rendkívül gyakran romlottak el, ezért — ha jól emlékszem — másfél óránként le kellett állni, meg kellett várni a lehűlést és az ellenőrzést. Gépi időt nagyon nehezen kaphatott az ember, és hosszú idővel előre kellett foglalni. A sok hiba miatt naponta többször volt karbantartás. A program beadása lépésenként haladt kapcsolósor beállításával, mint kettes számrendbeli egységek beállítóival. A megbízhatóság növelésére azt a módszert alkalmaztuk, hogy részekre bontottuk a programot, majd a részeket önálló programként futtattuk, az egyes részek eredményeit feljegyeztük, és a következő rész futtatásánál kézzel beadtuk. A részeredmények jóságát úgy biztosítottuk, hogy minden részen háromszor ment végig a program, és ha kétszer azonos eredményt kapott, akkor ezt elfogadta. (Munkánkat az Akadémia annyira sikeresnek ítélte, hogy különjutalomban részesültünk.) Mivel a programokat gépi kódban elemi lépésekre lebontva kellett írni, de a memória nagyon kicsi volt, ezért a programozó nagyon igényesen fogalmazta meg a feladatot, és rengeteg ötletre volt szüksége a megvalósításhoz, vagyis a programkészítés technikailag, tudományosan igen pallérozott virtuózt kívánt. Interjúalanyom tehát szakmánknak már akkor művésze volt. Az elsőként említett (FORTRAN-szerű) fordítóprogramot követő programkészítői tevékenységéről már az interjúban lesz szó.

Kérdés: Ön úgy ismert, mint egyike a legjobb programozóknak. Melyik program alapján vélik ezt?

Válasz: Egy sor körülménynek köszönhetően részt vettem a Xerox PARC kutatómunkájában a mikroszámítógép-forradalom korai korszakában. Egy csoportot vezettem itt, amelyik a BRAVO dokumentum-processzort fejlesztette. Ez volt az egyik legelső olyan szerkesztő, amely az „amit Ön lát, az történik" szemléleten alapult. Amidőn a Microsoft céghez kerültem, az én csoportom készítette a Multiplan, Microsoft Word, Microsoft Chart, Microsoft File, Microsoft Excel és más népszerű alkalmazói programokat. És szintén résztvettem a „grafikus felhasználói interfész" népszerűsítésében, különösen az „egér" alkalmazásában.

Kérdés: Jellemzi-e a program a készítőjét? Képes lehet-e valaki, természetesen szakember, felismerni a programról a szerzőt?

Válasz: Ez egy nagyon érdekes kérdés. A forráslistákat rendszerint aláírják a szerzők a dokumentálás, a védjegyezés szempontjából, és ez ad is kisfokú védelmet a nem engedélyezett másolás ellen. Természetesen a forráslisták titkot képeznek a kívülállók számára. Ha nekem megmutatnak egy nem jelzett forráslistát, a legtöbb, amit meg tudok határozni, a „tájszólás": mint pl. Unix-, Multix-, Tenexstílus. Bármely kód felismerésére képes vagyok, ha az én mostani vagy korábbi csoportjaim csinálták, mert egyezményes jelölésmódunk van.

A programozó nevének feltüntetése a terméken, ahol a vásárló is látja, már más dolog. Az Electronic Arts, egy kaliforniai szoftverkiadó olyan messzire megy a szerző (Bill Budge) reklámozásában, mintha egy rocksztár lenne. Bill képe rajta van a programcsomagon. Két dolog fontos ezzel kapcsolatban: Bill tényleg úgy néz ki, mint egy rocksztár, és a szóban forgó játékprogramokat többé-kevésbé segítség nélkül írta. Sok Macintosh-program bejelentkezéskor kiírja a szerző nevét (pl. Bill Atkinson). Az APPLE beépít a menübe egy parancsot, ami a program rövid leírását adja, és ez többek közt megnevezi a programot, tartalmazza a változat sorszámát, a védjegyet és a szerző nevét. A Microsoft-terméket általában nagy csoportok készítik, ezért mi nem jelezzük a programozó nevét. Általában a programozók elutasítják azt a megoldást, hogy bizonyos titkos parancsokat építsenek be a programba, amelyeket aktiválva megtudható legyen, ki a szerző.

Kérdés: A BASIC marad a legelterjedtebb nyelv az olcsó személyi számítógépeknél, vagy valamilyen másik? Ha igen, melyik?

Válasz: Én úgy vélem, hogy a BASIC, ezenbelül a Microsoft BASIC marad az elsődlegesen kedvelt nyelv nemcsak a legolcsóbb, hanem a legdrágább „irodai automatizálási" rendszereknél is. A BASIC-et folyamatosan kiterjesztik, így teljesítőképességében és jellemzőiben versenyképes marad. Gyakran hallani olyan tényeket, amelyek favorizálják a PASCAL-t, SMALLTALK-ot, LOGO-t. Úgy vélem, ezek csak a felhasználók egy-egy csoportját érintik. Illusztrációként említem az elektromos számológépek korai korszakának nagy kérdését: mit használjunk, "lengyel" vagy „algebrai" rendszert? A „lengyel" rendszert megérteni sokkal keservesebb egy átlagembernek, de azok a tudósok, akik könnyen elsajátították, lelkesen ajánlották mindenkinek, bár ezek közül a többség természetesen képtelen volt az értelmezésére. Nyilván az „algebrai" rendszer győzött. Ugyanezért keveseknek sikerült élvezni a PASCAL vagy a SMALLTALK jótéteményeit, mert terhelve vannak megértési nehézségekkel.

Kérdés: Milyennek látja az „ikonos" operációs rendszerek jövőjét?

Válasz: A Microsoft úgy véli, hogy a jövő a grafikus felhasználói interfészé (Graphical Users Interface — GUI.) Ez nemcsak az „ikonokat" jelenti, hanem minden bittérkép szerkezetű képernyővel dolgozó interfészt. Ilyenek pl. azok az interfészek, amelyek „egeret" használnak „a válaszd ki a művelet jelképét és hajtsd végre a műveletet" típusú parancsokhoz, továbbá, amelyek görgetési technikával kezelik a menüt, vagy ide tartoznak azok is, amelyeknél az almenük ún. dialógusmezők — és számos más is. Cégünk segítette az APPLE-éket abban, hogy ezeket az ötleteket a Macintosh-sal piacra vihessék. Most azzal foglalkozunk, hogy MS-DOS-gépeken alkalmazzuk a GUI-t. Ezt az operációs környezetet nevezik Microsoft Windows-nak. A legtöbb alkalmazói programunk már erre épül.

Kérdés: Számít valami alapvető változásra a számítógépek szerepének növekedésében?

Válasz: Az elkövetkező 5 évben nem. Mi arra számítunk, hogy a *személyi számítógépek odakerülnek minden iroda és otthon asztalára az Egyesült Államokban.* Ezt elősegíti: az egész országra kiterjedő elektronikus posta, zene a MIDI-interfészen keresztül (ez egy szabványosított bemenet), a mozifilmkészítés az új szuperszámítógépekkel. Alapvető változást csak a Artifical Intelligence /AI/ (mesterséges intelligencia) rendszerek elterjedése hozhat. Ezek egy része valószínűleg csalódást okoz majd az elvárásokhoz képest, de lesznek olyanok is, amelyek nagyon jól beválnak. Cégünknél ezeket „Softer Software"-nek nevezzük, szembeállítva a hagyományos szoftverek „lágyságnélküliség"-ét a jövőbeliek rugalmasságával. Ez azt jelenti, hogy a szoftver képes változtatni a formáját, fejleszteni a kapcsolatát a felhasználóval, ahogy a felhasználó ismeretei bővülnek. Ezek közül sok képes lesz arra, hogy magába fogadja az alkalmazási területén folyamatosan változó ismereteket. E szoftverek programnyelvei „szabály alapúak" lesznek.

Kérdés: Mennyire tartja fontosnak a CD-ROM-okat? Hol tart ma a fejlesztésük? Mi a Microsoft szerepe ebben?

Válasz: A CD-ROM tárolóképessége hihetetlenül nagy, olyan nagy, mint egy könyvsorozaté! Képzeljük el, hogy ma már előállítható egy kis térfogatú, törhetetlen csomag — egyetlen kemény kötésű könyv áráért —, ami géppel olvasható, és a benne tárolt információ egyenértékű néhány enciklopédiával. Két dolog fontos itt: a *nagyon alacsony ár és a hihetetlenül nagy tárolókapacitás!* Ami szintén nagyon fontos: a fejlesztési költséget megfizetik a zeneszerető vásárlók! Cégünk központi szerepet vállal a fejlesztésben. A megoldás kulcsa ugyanis itt is a szoftver, mind a felvételkészítésben, mind a lejátszásban. Mi szerveztük az I. CD-ROM Világkonferenciát, és szervezzük a másodikat ebben az évben. Egy új szervezeti egységet állítottunk fel, a CD-ROM technológia részleget. Mi már bejelentettük az MS-DOS által támogatott CD-ROM-ot, és sok más új termékünk van fejlesztés alatt.

Kérdés: Az MSX rendszer nem aratott sikert a felhasználóknál. Várható-e egyáltalán általánosan elfogadott hardver-szoftver rendszer kialakulása?

Válasz: Az MSX egy körülhatárolható és korábban kaotikus piacra, a 8 bites házi számítógépek japán piacára készült. Az eredmény igen jó volt, hiszen több mint egymillió ilyen gépet adtak el. Természetesen a 8 bites gépek nem sokáig jelentették a világszínvonalat. Az aszimptotikus megközelítése egy általánosan elfogadható hardver-szoftver rendszernek nemcsak kívánatos, de szükséges is mind a vásárlók, mind a szoftvergyártók számára. Az amerikai piacon minden jelzés szerint az Intel 386 architektura és az MS-DOS, Microsoft Windows következő generációja lesz a szabvány a professzionális célú személyi számítógépeknél. Várhatóan ez is, ahogy a korábbi esetekben is, nemzetközi sztenderddé válik. Az üzleti életben pedig a Macintosh architektúra lesz a mérték. *Az olcsó gépek piacát az MS-DOS-gépek fogják uralni.* A különleges igények kielégítésére továbbra is speciális gépeket fognak használni.

Kérdés: A szoftverfejlesztés rutinszerű tömegtermeléssé válik, vagy nagy egyéniségek meglepő ötletei lesznek a meghatározói?

Válasz: Sok kutató vélte és véli, hogy a a szoftverfejlesztés egyre inkább rutinná válik, ami a költségek csökkentését eredményezi. Egy bizonyos mértékig sok szoftver (mint pl. a könyvelő-, bérszámfejtő-, jelentéskészítő programok) fejlesztése mindig rutinmunka. *Ez nem igaz a milliós példányszámban eladott, a szoftverpiac listavezető termékeire.* Az *olyan termékek alkotói*, mint pl. a Microsoft BASIC, VisiCalc, Lotus 1—2—3, Microsoft Flight Simulator kevesen vannak, *egy-két különleges képességű és különlegesen keményen dolgozó ember*, akik a megoldás víziójától üzve, gyötörve, megszállottan dolgoztak, amíg csak el nem készült a mű. „Testhezálló" programot készíteni kissé hasonlít egy szívműtéthez. Ott is lehetetlen, hogy egy párnál több ember operáljon, de másoktól minden segítséget meg kell kapniuk. Másrészről egy mai korszerű szoftvertermléknek nagyon átfogónak kell lennie — pl. egy szövegszerkesztőnek az adatoknak a más termékkel történő cseréje céljából támogatnia kell különböző fájlformátumokat. Egy fájlkonvertáló program készítése még akkor is sok munkát jelent, ha jó technikát használunk. A programkészítési munkához hozzáadódik még a használati utasítások megírása és gyártása, a termék piacra vitele, termelése, továbbadása a forgalmazónak. Addig, amíg a program magjának megalkotása egy kis csoport műve marad, a végtermék készítésében már százak vesznek részt.

Kérdés: A számítógép-használat annyira egyszerűsödik, mint pl. a rádió, a tévé esetében, vagy szükséges lesz valamilyen számítógépi nyelv megtanulása?

Válasz: Ez kulcskérdés! Az embereknek — műszakiaknak és nem műszakiaknak — sok a téves elképzelésük a számítógépekről és a gépek teljesítményéről. Gyakran túlbecsülik a „gondolkodó gépek" képességeit, máskor nem veszik észre, hogy valami egy számítógépet tartalmaz — pl. egy modern automata fényképezőgép, amely erősen eltérő megvilágítási viszonyok közt is képes dolgozni. A kérdésre a válasz, hogy a számítógépek mindenkinek segíteni fognak. A fényképezőgépeki számítógép erősen leegyszerűsítette a képkészítést. *Az emberek mondjuk a Macintosh-sal való munkát olyan könnyen képesek megtanulni, mint egy írógéppel — gyorsan és bárki. Vagy mint ahogy egy zsebszámológép, könyvelőberendezés, rajztábla, rajzkatalogizáló eszköz alkalmazását könnyen megtanulhatják.* Ezeknek az eszközöknek a funkciói ugyanúgy állanak a felhasználó rendelkezésére egy számítógép által, mintha külön-külön eszközeink lennének. *Ilyenkor a számítógép mint grafikus felhasználói interfész megkönnyíti ezen eszközök kezelését. Programozói tudás ehhez nem szükséges.* Ha azonban ismerik a programozás elemeit, az többletet adhat. Sokkal hatékonyabban dolgozhat ez a személy a géppel: nemcsak egy betűt lesz képes beadni, hanem például egy levelező rutint is ír. A puszta könyvelésen túl el tudja készíteni a saját könyvelőprogramját is. A programozás alapeszméi — a tárolás, az utasításssorozatok — valóban egyszerűek, és ezek természetszerűen hozzá tartoznak az általános és középiskolák matematika tananyagához. *A programozás talán sokkal bonyolultabb valami, mint az osztás, de egyszerűbb, mint a négyzetgyökvonás!*

A fentiekhez csak rövid megjegyzést fűzök. Feltétlenül figyelembe veendő, hogy *ott az MS-DOS-gépek a házi számítógépek kategóriájába kerülnek, míg nálunk ezek a professzionális gépek csúcsai.* Fontosnak tartom, hogy a *kommersz programok készítése rutinmunka (aminek az árakban is tükröződnie kellene itthon is), igazán nagy sikerű terméket pedig nem átlagképességűek nagy csoportjai fejlesztenek, hanem különleges képességű egyedek (akiket ebben segíteni kellene, és rendesen megfizetni, ha eredményt érnek el). Gondolkozzunk, és utánozzuk, amit érdemes, mert ha ezt nem tesszük, csak nőhet az elmaradásunk!*

DR. SIMONYI ENDRE

VC20

Disassembler program

Ez a program VC20-tulajdonosnak készült, ha nincs bővítőjük, viszont már ismerkednek a gépi kódú programozással. Mivel a rendelkezésre állló terület mindössze 3,5 kbájt, a program írásánál elsődleges szempont volt a tömörség. Természetesen ez a többletszolgáltatások nélkülözésével járt. A program így is 890 bájt, amiből a táblázatok több mint 550 bájtot foglalnak el. A program leírása az *1. listában* megtalálható. Itt most csak a táblázatokról szólok.

Az *1. táblázatban* a program a gépi kódú utasítástól függően egy 0 és 168 közötti értéket olvas be. Ez adja meg a *3. táblázat* mnemonikjainak kezdőcímét. A *2. táblázat* kicsit bonyolultabb az előzőknél. Feladata a címzéskód megadása. Mivel összesen 12-féle címzésmód van, így tárolásához 4 bit is elegendő. Egy bájton tehát két címet tárolhatunk. Ezt mutatja a *4. táblázat*. Attól függően, hogy páros vagy páratlan az utasítás kódja, az alsó vagy a felső 4 bit szerepel az értelmezésnél.

A program a legtöbb fordítótól eltérően tízes számrendszerben dolgozik. Beírásához két segédprogramot ajánlok (*2.* és *3. lista*). Előtte azonban feltétlenül be kell állítani a BASIC tár végét: POKE55,134:POKE56,26.

Beírás után felvétele a következő:

POKE55,0:POKE56,31

POKE43,134:POKE44,26 a program eleje

POKE44,248:POKE46,29 a program vége+1

majd SAVE"DISASSEMBLER",1,1 és a VERIFY—jal ellenőrizzük.

Beolvasás a LOAD"DISASSEMBLER",1,1 vagy LOAD"",1,1-gyel kezdeményezhető, az indítás pedig a SYS 6790 paranccsal.

Indítás után egy kérdőjel jelenik meg. Ide írjuk azt a címet, ahonnan a fordítás kezdődjön. Az F7 billentyű nyomva tartásával a disassemblálás folyamatos. F1-re új címtől kezdődhet a fordítás. A CTRL lenyomva tartásával a fordítás lassítható.

KOZMA TIBOR

```
6790  JSR/A  52221      32,253,203    ;  INPUT rutin
6793  LDX  # 0          162,0         ;  a szám tárolása a 89-90
6795  STX/N  89         134,89        ;  címeken. Kezdőérték nulla
6797  STX/N  90         134,90        ;
6799  LDA/AX 512        189,0,2       ;  pufferből ASCII karakterek
6802  BEQ    6848       240,44        ;  beolvasása. Vége?
6804  SBC  # 47         233,47        ;  minusz"0" kódja-1
6806  STA/N  7          133,7         ;  tizes számrendszerbeli szám
6808  LDA/N  90         165,90        ;  kiszámítása
6810  STA/N  34         133,34
6812  LDA/N  89         165,89
6814  ASL               10            ;  szorzás tizzel
6815  ROL/N  34         38,34
6817  ASL               10
6818  ROL/N  34         38,34
6820  ADC/N  89         101,89
6822  STA/N  89         133,89
6824  LDA/N  34         165,34
6826  ADC/N  90         101,90
6828  STA/N  90         133,90
6830  ASL/N  89         6,89
6832  ROL/N  90         38,90
6834  LDA/N  89         165,89
6836  ADC/N  7          101,7
6838  STA/N  89         133,89        ;  és betevése a 89-90 címre
6840  BCC    6844       144,2
6842  INC/N  90         230,90
6844  INX               232
6845  JMP/A  6799       76,143,26     ;  következő karakter
6848  LDA  # 13         169,13        ;  soremelés
6850  JSR/A  65490      32,210,255    ;  CHROUT rutin
6853  LDA/N  197        165,197       ;  billentyüzet lekérdezése
6855  CMP  # 39         201,39        ;  F1 gomb ?
6857  BEQ    6790       240,187       ;  igen, vissza az INPUT-hoz
6859  CMP  # 63         201,63        ;  F7 gomb ?
6861  NOP               234
6862  NOP               234
6863  NOP               234
6864  BNE    6853       208,243       ;  nem, ujra kérdezze a billentyüt
6866  LDA/N  90         165,90        ;  akkumulátorba a hight byte
6868  LDX/N  89         166,89        ;  X-be a low byte
6870  JSR/A  56781      32,205,221    ;  a cím kiirása
6873  JSR/A  7062       32,150,27     ;  szünet rutin
6876  LDY  # 0          160,0         ;
6878  LDA/IY 89         177,89        ;  a címbájt értékének a beolvasása
6880  PHA               72
6881  TAX               170
6882  LDA/AX 7080       189,168,27    ;  utasításkód beolvasása
6885  TAX               170
6886  LDA/AX 7464       189,40,29     ;  mnemonik beolvasása
6889  JSR/A  65490      32,210,255    ;  és kiirása
6892  INX               232
6893  INY               200
6894  CPY  # 3          192,3         ;  mind a három betü kiirva ?
6896  BNE    6886       208,244       ;  nem, a köv. betü beolvasása
6898  NOP               234
6899  NOP               234
6900  NOP               234
6901  PLA               104           ;  utasitás kódja az A-ba
6902  PHA               72
6903  LSR               74            ;  osztás kettővel
6904  TAX               170
6905  LDY/AX 7336       188,168,28    ;  címzésmód beolvasása
6908  PLA               104           ;  utasitás kódja ismét az A-ba
6909  AND  # 1          41,1          ;  páros ?
6911  BEQ    6919       240,6         ;  igen
6913  TYA               152           ;  nem,
6914  AND  # 15         41,15         ;  csak az alsó 4 bit kell
6916  JMP/A  6924       76,12,27
6919  TYA               152           ;  páros
6920  LSR               74            ;  csak a felső 4 bit kell
6921  LSR               74
6922  LSR               74
6923  LSR               74
```

1. lista

```
10 FORT=6790TO7463
20 INPUTA:POKET,A
30 PRINTT,A:NEXT
```

2. lista

1. táblázat

```
7080 : 90 144   0   0   0 144 129   0 123 144     7210 :  0   0  18  12  15   0  69   0  33   0
7090 : 129  0   0 144 129   0 188 144   0   0     7220 : 18  12  15   0  93  12   0   0  18  12
7100 :  0 144 129   0  51 144   0   0   0 144     7230 : 15   0  36  15  39   0   0  12   0   0
7110 : 129  0  81 141   0   0 158 141 132   0     7240 :  9   3   6   0   9   3   6   0  38   3
7120 : 126 141 132   0 158 141 132   0 185 141   7250 : 27   0   9   3   6   0  96   3   0   0
7130 :   0   0   0 141 132   0  45 141   0   0   7260 :  3   3   6   0  57   3  42   0   9   3
7140 :   0 141 132   0  87 147   0   0   0 147   7270 :  6   0 165 159   0   0 165 159  24   0
7150 : 135   0 117 147 135   0  78 147 135   0   7280 : 75 159  66   0 165 159  24   0 102 159
7160 : 111 147   0   0   0 147 135   0  60 147   7290 :  0   0   0 159  24   0  54 159   0   0
7170 :   0   0   0 147 135   0  84 153   0   0   7300 :  0 159  24   0 162 156   0   0 162 156
7180 :   0 153 138   0 120 153 138   0  78 153   7310 : 21   0  72 156 168   0 162 156  21   0
7190 : 138   0 114 153   0   0   0 153 138   0   7320 : 99 156   0   0   0 156  21   0  48 156
7200 : 63 153   0   0   0 153 138   0   0  12   7330 :  0   0   0 156  21
```

```
6924  STA/N  1                133,1          ;  elmenteni az I bájtba
6926  ASL                     10             ;  A=Ax2 kiszámítása
6927  CLC                     24
6928  ADC/N  1                101,1
6930  TAX                     170
6931  LDY    # 0              160,0
6933  LDA/AX 7635             189,211,29     ;  címzésmód kiírása
6936  JSR/A  65490            32,210,255
6939  INX                     232
6940  INY                     200
6941  CPY    # 3              192,3          ;  kiírva ?
6943  BNE    6933             208,244        ;  nem, köv. karakter
6945  JSR/A  7067             32,155,27      ;  szünet rutin
6948  LDY    # 0              160,0
6950  LDA/N  1                165,1          ;  címzésmód A-ba
6952  CMP    # 0              201,0          ;  beleértett ?
6954  BEQ    6982             240,26
6956  CMP    # 1              201,1          ;  közvetlen ?
6958  BEQ    6988             240,28         ;
6960  CMP    # 2              201,2          ;  relatív ?
6962  BEQ    7004             240,40
6964  CMP    # 8              201,8          ;  nullás ?
6966  BMI    6988             48,20          ;  egyik sem, akkor abszolut
6968  JSR/A  7073             32,161,27      ;  olvasd be a cim low bájtját
6971  LDA/IY 89               177,89         ;
6973  TAX                     170            ;  tedd az X-be
6974  JSR/A  7073             32,161,27      ;  következő bájt
6977  LDA/IY 89               177,89         ;  beolvasása az A-ba
6979  JSR/A  56781            32,205,221     ;  és kiírása
6982  JSR/A  7073             32,161,27      ;  köv. bájt
6985  JMP/A  6848             76,192,26      ;  új utasitás következik
6988  JSR/A  7073             32,161,27      ;  nullás cimzés. a köv. bájt
6991  LDA/IY 89               177,89         ;  beolvasása
6993  TAX                     170            ;  X-be tétele
6994  TYA                     152            ;  a hight byte nulla
6995  JSR/A  56781            32,205,221     ;  szám kiirása
6998  JSR/A  7073             32,161,27      ;  köv. bájt
7001  JMP/A  6848             76,192,26      ;  új utasitás jön
7004  JSR/A  7073             32,161,27      ;  relatív cimzés
7007  LDA/IY 89               177,89         ;  köv.bájt A-ban
7009  STA/N  205              133,205        ;  és mentése a 205. bájtba
7011  JSR/A  7073             32,161,27      ;  köv. bájt
7014  LDA/N  89               165,89         ;  a szám beirása a 90-91 cimekre
7016  STA/N  91               133,91
7018  LDA/N  90               165,90
7020  STA/N  92               133,92
7022  LDA/N  205              165,205        ; ugrúscim A-ba
7024  BMI    7040             48,14          ; negativ ?
7026  LDA/N  91               165,91         ; ugrúscim kiszámitása
7028  CLC                     24
7029  ADC/N  205              101,205
7031  STA/N  91               133,91
7033  BCC    7037             144,2
7035  INC/N  92               230,92
7037  JMP/A  7052             76,140,27
7040  DEC/N  205              198,205        ;  vond ki a 91-92-es cimekből
7042  LDA/N  91               165,91
7044  ADC/N  205              101,205
7046  STA/N  91               133,91
7048  BCS    7052             176,2
7050  DEC/N  92               198,92
7052  LDA/N  92               165,92
7054  LDX/N  91               166,91
7056  JSR/A  56781            32,205,221     ;  kapott cim kiirása
7059  JMP/A  6848             76,192,26      ;  uj utasitás
7062  LDA    # 32             169,32         ;  SZÜNET RUTIN
7064  JSR/A  65490            32,210,255     ;  space kiirás
7067  LDA    # 32             169,32         ;
7069  JSR/A  65490            32,210,255     ;  még egy space
7072  RTS                     96
7073  INC/N  89               230,89         ;  növelő rutin
7075  BNE    7079             208,2
7077  INC/N  90               230,90
7079  RTS                     96
```

```
10 FORT=7464TO7671STEP3
20 PRINTCHR$(13);T,
30 FORS=0TO2
40 GETA$:IFA$=""THEN40
50 POKET+S,ASC(A$):PRINTA$;
60 NEXT:NEXT
```

3. lista

```
7336 :   6   0   3  48   1   7   8   0 135   0
7346 :   4  64  10   0   9 144 134   0  51  48
7356 :   1   0 136   0  39   0   4  64  10   0
7366 :   9 144   6   0   3  48   1   0 136 128
7376 :   7   0   4  64  10   0   9 144   6   0
7386 :   3  48   1   0 184 128   7   0   4  64
7396 :  10   0   9 144   6   0  51  48   0   0
7406 : 136 128  39   0  68  80  10   0   9   0
```

```
7416 :  22  16  51  48   1   0 136 128  39   0
7426 :  68  64  10   0 153 160  16   0  51  48
7436 :   1   0 136 128  39   0   4  64  10   0
7446 :   9 144  27   0  51  48   1   0 136 128
7456 :  39   0   4  64  10   0   9 144
```

2. táblázat

3. táblázat

0 .	7464	***
3 .	7467	LDA
6 .	7470	LDX
9 .	7473	LDY
12 .	7476	STA
15 .	7479	STX
18 .	7482	STY
21 .	7485	INC
24 .	7488	DEC
27 .	7491	TAX
30 .	7494	TAY
33 .	7497	TXA
36 .	7500	TYA
39 .	7503	TXS
42 .	7506	TSX
45 .	7509	SEC
48 .	7512	SED
51 .	7515	CLC
54 .	7518	CLD
57 .	7521	CLV
60 .	7524	CLI
63 .	7527	SEI
66 .	7530	DEX
69 .	7533	DEY
72 .	7536	INX
75 .	7539	INY
78 .	7542	JMP
81 .	7545	JSR
84 .	7548	RTS
87 .	7551	RTI
90 .	7554	BRK
93 .	7557	BCC
96 .	7560	BCS
99 .	7563	BEQ
102 .	7566	BNE
105 .	7569	BMI
108 .	7572	BPL
111 .	7575	BVC
114 .	7578	BVS
117 .	7581	PHA
120 .	7584	PLA
123 .	7587	PHP
126 .	7590	PLP
129 .	7593	ASL
132 .	7596	ROL
135 .	7599	LSR
138 .	7602	ROR
141 .	7605	AND
144 .	7608	ORA
147 .	7611	EOR
150 .	7614	BIT
153 .	7617	ADC
156 .	7620	SBC
159 .	7623	CMP
162 .	7626	CPX
165 .	7629	CPY
168 .	7632	NOP

4. táblázat

Egy bájt:

7	6	5	4	3	2	1	0	bitek

páros | páratlan

0 –	beleértett	– 0	7635.	___
16 –	közvetlen	– 1	7638.	#
32 –	relatív	– 2	7641.	___
48 –	nullás	– 3	7644.	/N_
64 –	nullás,X	– 4	7647.	/NX
80 –	nullás,Y	– 5	7650.	/NY
96 –	ind. indirekt	– 6	7653.	/IX
112 –	ind. index	– 7	7656.	/IY
128 –	abszolút	– 8	7659.	/A_
144 –	abszolút,X	– 9	7662.	/AX
160 –	abszolút,Y	– 10	7665.	/AY
176 –	indirekt	– 11	7668.	/I_

_ = SPACE

Az ábra jobb oldalán a program utolsó bájtjainak az értéke látható. Ezeket a jelöléseket használja a program. Ha valakinek ezek nem felelnének meg, akkor bármilyen 3 bájtot foglaló jelölésre átírhatja őket.

Különleges input

COMMODORE 64

A rutin csak olyan billentyűk lenyomását fogadja el, melyeket az idézőjelek között megadtunk, s a változó értéke a lenyomott billentyű sorszáma lesz.

A rutin legelőnyösebb felhasználási lehetősége: menüknél annak eldöntése, hogy melyik „fogást" választotta a felhasználó. Ennek bemutatására szolgál a BASIC példaprogram (*1. lista*), mely a betöltőprogramot is tartalmazza a magyarázatokkal együtt. Az A$=CHR$ (PEEK(880)) értékadással megkaphatjuk a karaktert is.

Az assembler program *(2. lista)* ismertetése:

120—180 A használt ROM-rutinok címe
200 A program a kazettapufferba kerül
220 Vessző átlépése
230—260 A programmutató átmásolása a $FB, $FC címre
270—280 Idézőjel vizsgálata
300—320 A két idézőjel közötti karakterek átlépése
330 Idézőjel átlépése
340 Vessző átlépése
350—370 Változó keresése, címének elmentése
390—400 Várakozás billentyű lenyomására
410 Megszakítás letiltása
420—430 A lenyomott billentyű ASCII kódjának elmentése
440—500 Az idézőjelek között megadott karakterek beolvasása és összehasonlítása a lenyomott billentyű ASCII kódjával
510—520 Ha nem egyezett meg, megszakítás engedélyezése, vissza a várakozó ciklusba
540—550 Ha megegyezett, akkor Y átalakítása lebegőpontos formátumba
560 Változó értékadása
570 Megszakítás engedélyezése
580 Visszatérés a BASIC-be

CSIZMAZIA TIBOR

1. lista

```
100 REM       K U L O N L E G E S   I N P U T
110 :
120 REM       KESZITETTE : CSIZMAZIA TIBOR
130 :
140 REM       HIVASA: SYS828,"12AB■",A
150 REM       EREDMENY: A=A LENYOMOTT BILLENTYU SORSZAMA.
160 REM       PL.:"B" LENYOMASARA A=4
170 REM       CSAK AZ IDEZOJELEK KOZOTT MEGADOTT
180 REM       BILLENTYUKRE REAGAL!
190 :
200 DATA 32,253,174,165,122,133,251,165,123,133
210 DATA 252,169,34,32,255,174,32,115,0,201
220 DATA 34,208,249,32,115,0,32,253,174,32
230 DATA 139,176,133,73,132,74,165,198,240,252
240 DATA 120,32,180,229,141,112,3,160,1,177
250 DATA 251,201,0,240,8,200,201,34,208,245
260 DATA 88,240,229,169,0,32,145,179,32,208
270 DATA 187,88,96
280 :
290 REM       --- BEOLVASAS ---
300 :
310 S=0:FORI=828TO828+72
320 READA:POKEI,A:S%=S%+A:NEXT
330 IFS%=9962THEN380
340 PRINT"*** DATA HIBA ***":END
350 :
360 REM       --- PELDA ---
370 :
380 PRINT"■■■■■■";TAB(12);"----------------
390 PRINT"■■";TAB(17);"■F1■ RUN
400 PRINT"■■";TAB(17);"■F3■ LIST
410 PRINT"■■";TAB(17);"■F5■ NEW
420 PRINT"■■";TAB(12);"----------------
430 PRINT"■■■";TAB(12);"KEREM VALASSZON!"
440 SYS828,"■■■",A
450 ONAGOTO460,470,480
460 RUN
470 LIST
480 NEW
```

2. lista

```
KARGET

1000                100         ;SYS828,"■■■",A
1000                110         ;
AEFD                120 CHKCOM  =$AEFD
0073                130 CHRGET  =$0073
B08B                140 VALKER  =$B08B
BBD0                150 VALTER  =$BBD0
E5B4                160 KARGET  =$E5B4
B391                170 INTFAC  =$B391
AEFF                180 IVIZSG  =$AEFF
1000                190         ;
033C                200         *=$033C
033C                210         ;
033C 20 FD AE       220         JSR CHKCOM
033F A5 7A          230         LDA $7A
0341 85 FB          240         STA $FB
0343 A5 7B          250         LDA $7B
0345 85 FC          260         STA $FC
0347 A9 22          270         LDA #"""
0349 20 FF AE       280         JSR IVIZSG
034C                290         ;
034C 20 73 00       300 TOVABB  JSR CHRGET
034F C9 22          310         CMP #"""
0351 D0 F9          320         BNE TOVABB
0353 20 73 00       330         JSR CHRGET
0356 20 FD AE       340         JSR CHKCOM
0359 20 8B B0       350         JSR VALKER
035C 85 49          360         STA $49
035E 84 4A          370         STY $4A
0360                380         ;
0360 A5 C6          390 VAR     LDA $C6
0362 F0 FC          400         BEQ VAR
0364 78             410         SEI
0365 20 B4 E5       420         JSR KARGET
0368 8D 70 03       430         STA KAR+1
036B A0 01          440         LDY #$01
036D B1 FB          450 HAS     LDA ($FB),Y
036F C9 00          460 KAR     CMP #$00
0371 F0 08          470         BEQ MEGVAN
0373 C8             480         INY
0374 C9 22          490         CMP #"""
0376 D0 F5          500         BNE HAS
0378 58             510         CLI
0379 F0 E5          520         BEQ VAR
037B                530         ;
037B A9 00          540 MEGVAN  LDA #$00
037D 20 91 B3       550         JSR INTFAC
0380 20 D0 BB       560         JSR VALTER
0383 58             570         CLI
0384 60             580 VEGE    RTS
0385                590         .END

 LINES:50   SYMBOLE:13   ERRORS:0

CHKCOM=AEFD  CHRGET=0073  HAS    =036D
KARGET=E5B4  MEGVAN=037B  TOVABB=034C
VEGE  =0384
INTFAC=B391  IVIZSG=AEFF  KAR    =036F
VALKER=B08B  VALTER=BBD0  VAR    =0360
```

Printerszalag – házilag

● Az Alpha-Micro számítógépekhez használatos LogAbax nyomtatók szalagjait, amikor elkopnak, egyszerűen ki kellene cserélni, ha vásárolható lenne a hozzávaló — papírdobozban elhelyezett — Olivetti festékszalag. De mint a legtöbb más végtelenített festékszalag, általában ez sem kapható. Átmenetileg ajánljuk az alábbi kisegítő megoldásokat. Az első a drágább, de aki sokszor és sokat nyomtat, valószínűleg ezzel fog élni.

A Commodore 64 MPS-802 nyomtatóhoz 1500,— Ft-ért kínálnak végtelenített szalagot, amelynek szélessége megegyezik az Olivetti festékszalag szélességével (15 mm). Ez a műanyag dobozból kis ügyességgel áthelyezhető a LogAbax műanyag dobozába (nem papírdobozba). Az Olivetti kiszerelésben a szalag Möbius módon van végtelenítve. A LogAbax nyomtató a szalag automatikus átfordulása után a felső helyett a szalag alsó részét koptatja, és így tovább, tehát a középső részét nem. Az MPS-802 a szalag középső részére ír, a felső és alsó részét nem érinti. Egy Olivetti szalag ilyenformán a két nyomtatón különkülön, tehát kétszer volna felhasználható. Az MPS-802 szalagja viszont nem Möbius végtelenítés, ezért a LogAbax csak a szalag felső részét „gyötri", de kézi szalagfordítással az alsó részét is hasznosíthatjuk. Lehetőleg pamutszalagot válasszunk! Ha csak celluloidszalag van, akkor az ajánlott sorrend: LogAbax (itt szalagfordítás), majd MPS-802. Celluloidszalagnál figyeljünk rá, hogy egy sávot ne nagyon koptassunk el, mert ez a szalag elég gyenge minőségű, új korában is könnyen szakad. A celluloidszalag terhelhetősége kb 1/10-e a pamutszalagénak, és szakadásának oka a LogAbax nyomtatófejének túlmelegedése is lehet (többpéldányos papír esetén). Ilyen munkáknál célszerű a nyomtatást úgy megszervezni, hogy a nyomtatófej le tudjon hűlni. Újabban celluloidszalagok vannak inkább forgalomban, így a nyomtatás továbbra is költséges marad. Szükség esetén mégis aránylag kényelmes az ajánlott szalagcsere mint helyettesítő megoldás. Az MPS-802 szalag áthelyezésére az eljárás a következő:

— Nyissa fel a LogAbax műanyag szalagszelencéjét!
— Emelje ki a papírdobozt; a vezetőgörgők felé eső részen vigyázzon, hogy a szalag ki ne szaladjon, ezt ujjal meg lehet gátolni.
— Gyömöszölje be a papírdobozba a szalagot, és zárja le a megmaradt füllel! (Ez a szalag az MPS-802 nyomtatóban még átfűzéssel felhasználható.)
— A LogAbax közvetlen közelében nyissa fel az MPS-802 műanyag dobozát (a felső rész fedele leszedhető), óvatosan, hogy a vezetőgörgők ki ne ugorjanak.
— Szedje ki a vezetőgörgőket és rugókat! Begyakorlott, ügyes kézmozdulattal áthelyezhető a felhajtogatott szalagmennyiség, de eleinte 10-ből 9 esetben biztosan szétugrik (főleg a celluloidszalag). Ilyenkor ismételje meg a műveletet!
— Az áthelyezés után zárja le a LogAbax szalagszelencéjét, és fűzze be a szokásos módon a szalagot. (Celluloidszalagnál ügyeljen a festékes oldalára.)

Ha az áthelyezés után szétugrik a szalag, akkor egy nagyobb dobozban kényelmesen összegyűjtve, nehogy összecsomózódjon, tudjuk a LogAbax szelencéjébe befűzni. A szalagot be kell helyezni a szelencébe, mintha az egyenesen átfutna rajta, majd befűzni. A szalag nagyobb része a printeren kívüli dobozban helyezkedik el, a printer műanyag fedelét lehajtva — akár nyomtatás közben is — automatikusan a szelencébe gyűjti a szalagot. Kézzel (pl. filctollal) beforgatható a szalag a szelencébe. Az egész művelet maximum 15 percet vesz igénybe. Az MPS-802 dobozába való befűzéshez már nagyobb kézügyességre van szükség.

Az olcsóbb megoldásnak is reméljük, sokan örülnek. Az 53,— Ft-ért vásárolható írógépszalag a nyomtatószalaggal azonos hosszúságú. A 16 mm-es írógépszalag teljesen megfelel a céljainknak, és így jóval olcsóbb megoldás. Mint nyomtatószalag a pamutszalag tartósabb is, és jobban kihasználható akármilyen más szalagnál, de nincs végtelenítve. Aki keveset nyomtat, kis odafigyeléssel még végtelenítés nélkül is használhatja.

Sajnos, jó végtelenítéseket nem sikerült kialakítanunk, a ragasztások sem váltak be. Még mindig kísérletezünk különböző módszerekkel, de hátha! — ötleteket várunk ötletért. (Az újságban természetesen vállaljuk az ötlet közvetítését. A szerk.)

S végül egy tanácsnak is fölfogható megjegyzés: ne tévesszen meg senkit, hogy a nyomtatószalag-szelencében a szalag szépen van összehajtogatva, ezt a vezető fogaskerekek elvégzik. Saját magunk nem leszünk eredményesek a szalag kézi összehajtogatásával. Pamutszalagnál nagyon lényeges, hogy a szalagon ne legyen sérülés, mert az megsértheti a nyomtatófejet.

Jó munkát!

Balla Lajos

THE USERS PORT

San Fernando Valley Commodore Users Group

A lemezár rejtélye

Mitől függ az üres lemezek ára? Vegyük szemügyre az ezt legjobban befolyásoló tényezőket.

Az egyoldalúság (single-sided)

Bizonyára sokaknak okozott fejtörést, vajon használhatják-e a „single-sided" lemezek mindkét oldalát. Gyakran hallani azt a véleményt, hogy ezeknek a lemezeknek csak az egyik oldaláért vállalja a gyártó cég a garanciát, mert a másik oldal gyengébb minőségű. Ez hihetőnek hangzik, ugye? Csökkenti az előállítási költségeket, ha a lemeznek csak az egyik oldalát ellenőrzik. Egyoldalú lemez is lemez . . . De éppen ez a dolog buktatója! A gyártó ugyanis nem tudhatja előre, hogy milyen típusú meghajtóba kerül majd a lemez. Például a Commodore lemezmeghajtó a lemezek hátsó oldalára ír, de nem ez a szabvány. Vannak olyan „single-sided" meghajtók, melyek a lemeznek csak az egyik oldalára írnak; más típusok a másik oldalra is írhatnak. Emiatt végső soron a gyártó cégeknek a lemezek mindkét oldalát egyenlő gondossággal kell elkészíteniük.

A kétoldaliság (double-sided)

Miért ne fordíthatnánk meg a lemezt, és használhatnánk a másik oldalát is? A 13,2 × 13,2 cm-es lemez szélén, a tetejétől számítva mintegy 3 cm távolságban kis téglalap alakú kivágás van. Ez teszi lehetővé a lemezre írást; ha a nyílást leragasztjuk, a lemez írásvédetté válik.

A másik oldalt is ki kell tehát vágni a lemez tetejétől számítva pontosan olyan távolságra, majd a kivágott lemezt megfordítva, a meghajtóba téve, azonosan dolgozhatunk ezzel az oldallal is. A kivágás alakja nem is lényeges — kör vagy téglalap egyaránt jó —, csak a helye fontos, és hogy kb. 6 mm széles és 4 mm hosszú legyen. A kivágás helyét egy másik lemez segítségével jelölhetjük ki, ha azt megfordítjuk. A közönséges lyukasztóval jó eredményt kapunk, de az sem baj, ha a lyuk kicsit nagyra sikerül. Kevés a valószínűsége, hogy megsérül a lemez.

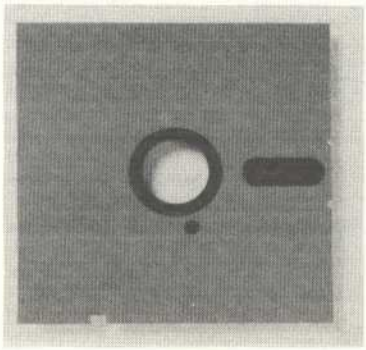
Sokan azt hiszik, hogy miközben a másik oldalra ír az egyoldalú meghajtó, elveszhetnek az eredeti oldalon tárolt adatok. Feltételezik, hogy a mágneses írófej olyan nyomást gyakorolhat írás közben a másik oldalra, ami az adatok megsemmisülését okozhatja. A fő ellenérv a tapasztalat: ilyesmi normális használat esetén még nem fordult elő. A meghibásodási arányt tekintve a statisztikák szerint nincs különbség az egyoldalú és a kétoldalú lemezek között.

A kétszeres sűrűség (double density)

Ez nagyon ravasz dolog. A vásárlók bolondot csinálnak magukból, mert nem tudják, hogy a kétszeres sűrűség valójában mit jelent. A kétszeres sűrűségű lemezek többe kerülnek, mert vastagabb rajtuk a mágneses réteg. De hogy jobbak-e? Ezek a lemezek arra valók, hogy a lemezmeghajtót erősebb mágneses jelek generálására tegyék képessé, mint amilyet a normál meghajtók generálni tudnak. Ez rendszerint akkor fontos, ha több bitet tárolunk inchenként, de teljesen szükségtelen normál meghajtók esetén. Sőt, mivel a normál meghajtó jele nem elég erős a dupla sűrűségű lemezhez, nagyobb a valószínűsége, hogy az értékes adatok elvesznek, ha mégis dupla sűrűségű lemezt teszünk fel.

A védőgyűrű

Vannak olyan lemezek, amelyeket védőgyűrűvel látnak el, és ezért drágábbak. A védőgyűrű kör alakú sáv a lemez belső szélén, amitől ez az él erősebb lesz. Fő célja, hogy megvédje a lemez szélét az összeroppanástól, amikor a meghajtó ajtaját úgy zárjuk le, hogy a lemez nincs teljesen a helyén. Ezt azonban jobb elkerülni, és kis figyelemmel, gondossággal nem is nehéz: lassan kell lezárni az ajtót, és azonnal kinyitni, ha ellenállásba ütközünk. A védőgyűrű problémát is okozhat: ha a lemez



hajlamos arra, hogy beleragadjon a borítójába, és ezért nem tudja elérni teljes forgási sebességét, a védőgyűrű csak fokozza a megcsúszás veszélyét.

A gyártó cég neve

Nem érdemes többet fizetni egy lemezért, csak mert nagy hírű cég gyártotta. Ha egy lemez átmegy a minőségi teszteken, teljesen mindegy, hogy melyik gyártótól való. A ráragasztott címkétől nem lesz jobb.

A tisztítókészlet

A meghajtó tisztításához nagyon óvatosan lássunk hozzá. Nem szabad bedőlni a hirdetésnek: „Vásároljunk tisztítófelszerelést a meghajtóhoz! Hetente legalább egyszer tisztítsuk meg az író/olvasó fejet!"

A szomorú valóság: mivel a tisztítófolyadék nagyon erős oldószer, csínján kell vele bánni. Ha a meghajtóval a tisztítás után azonnal dolgozunk is, a folyadék a fejről a lemezre kerül, és a lemez másik oldala tönkremegy. Ha minden lemeznek csak az egyik oldala a munkafelület, az oldószer nem okoz problémát. Természetesen tisztításra szükség van, de nem hetente, hanem 1—4 évente, és a műveletthez az egyszerű alkohol is megfelel. Ez nem károsítja sem a meghajtót, sem a lemezt.

A helyes bánásmód

A legtöbb lemezhibát a lemez helytelen gondozása és használata okozza. Sokkal többet, mint amiről a gyártó cégek tehetnek. Természetesen a lemezeket mágneses mezőktől távol kell tartani, tiszta és száraz helyen. Sohasem szabad olyan eszközzel írni a lemez címkéjére, amely megnyomhatja az alatta levő mágneses felületet. Általában óvni kell a lemezt a nyomástól.

Talán az egyetlen olyan véletlen lemezhiba, amelyet nem a meghajtó okoz, ha a lemez beragad a borítóba. Ezért tanácsosabb a lemezeket függőlegesen és nem vízszintesen tárolni. Minden nyomás, amely a lemez borítóját éri, leragadáshoz vezethet — beleértve a szoros tárolást, akár függőlegesen, akár vízszintesen —, melynek következtében a lemez nem tudja elérni teljes forgási sebességét. Ez megszakítási hibákat okoz.

Reméljük, hasznos tanácsokkal szolgált ez a írás, bár bizonyára vannak, akik nem értenek egyet vele. A legbiztosabb azt tenni, ami legjobban bevált.

Tom Flander nyomán írta:
PÁL MAGDA

Z80 szimulátor

Valószínűleg sokunkkal megesett már, hogy egy idegen programban látott megoldás vagy trükk felkeltette érdeklődésünket. Szerettük volna tudni, vajon hogyan valósította meg a programozó: esetleg egy az egyben beépítenénk saját programunkba az adott részletet. Máskor mindössze annyi a vágyunk, hogy a gép „agyába" belenézhessünk, hátha van benne valami olyan részlet, amely alkalmas egyes feladatok megoldásához.

Ezek a kívánságok nem mindig teljesíthetők egyszerűen. Ha a program BASIC-ben íródott és a lista hozzáférhető, vagy egy gépi kódú programról assembler listát tudunk szerezni (ami szinte lehetetlen), akkor viszonylag könnyű a dolgunk. De ha mindezek nincsenek, és csak egy gépi kódú programunk van, akkor tesz jó szolgálatot a Z80 SZIMULÁTOR program, amely alkalmas gépi kódú programok működésének végigkövetésére. A megfejtés közben könnyedén begyakorolhatók a Z80 mikroprocesszor gépi kódú utasításai, és a megfejtett programból ötleteket kaphatunk a gépi kódú programozáshoz. Előbb-utóbb sort kerítünk rá, hogy segítségével hibát keressünk idegen gépi kódú programban. Ehhez azonban már egyébként is némi gyakorlatra és tapasztalatra van szükség.

A program szimulálja a Z80 mikroprocesszor regisztereit, és a szimulált program gépi kódú utasításai szerint halad. A szimulált PC regiszter által mutatott memóriából beolvassa a tartalmát, és ezt gépi kódú utasításként értelmezve kiírja az utasítás jelentését. Ezután a szimulált regiszterekkel elvégzi az utasításban előírtakat. Az utasítások hatását — a regiszterérték-változásokat — a képernyőn követhetjük nyomon. A program egyesével megy végig a szimulált program utasításain, így minden lépés részletesen elemezhető.

A szimulátor használata

Ha a ROM-ban lévő programra vagyunk kíváncsiak, egyszerűen betöltjük a szimulátort, és a PC regisztert beállítjuk arra az értékre, ahonnan a szimulációt el akarjuk kezdeni (a PC beállítását lásd később). A címnek azonban „biztos forrásból" kell származnia, mert egyébként nem azt kapjuk eredményül, ami a valóság. Ennek hiányában sajnos a „kályhától" kell elindulni; ez a Z80 esetében a 0 cím. Innen kezdve viszont később már sok rutin helye ismert lesz.

Kicsit nehezebb a betölthető gépi kódú programok megfejtése. Ezeket ugyanis először el kell helyezni a memóriának abba a részébe, amely a szimulátor betöltése után is szabad marad: 64 kbájtos gépnél 39 500—58 500-ig terjed ez a terület. A szimulátor jelenlegi verziójával 48 kbájtos géppel csak a ROM fejthető meg. A 32 kbájtos géppel nem is használható ez a verzió. Ha a program eredetileg nem ezen a területen helyezkedett el, akkor azoknál az abszolút címes ugrásoknál és szubrutinhívásoknál, amelyek a szimulált program eredeti területére vonatkoznak, a cím értékét az eltolás mértékével szigorúan helyesbítsük! A helyesbítést az adott utasítás 2. lépésének végrehajtása után kell elvégezni (hogy melyik a 2. lépés, kicsit később kiderül). A program csak a szoftver szempontjából szimulálja a processzort, tehát az olyan programok megfejtése, amelyekben bizonyos megszakítások is szerepet játszanak és ezek bekövetkezésének ideje nem közömbös a főprogram szempontjából, igen nagy körültekintést igényel.

Egy program szimulálásához ismernünk kell a program kezdőcímét és a regiszterek belépési értékeit. Nem nagy baj, ha ez utóbbiakat nem ismerjük, mert a szimuláció során szinte biztosan kiderülnek a szükséges értékek, vagy az, hogy közömbös a belépési tartalom.

A program betöltése után a szimulált regiszterekben olyan kezdeti értékek keletkeznek, mint amilyenek a processzorban hardver RESET után lehetnek. Ezek a következők:

PC=0
I=0
R=0

A többi regiszter tartalma véletlenszerű lesz.

Egy utasítás szimulálása két lépésben halad (minden regiszterhivatkozás a szimulált regisztert jelenti).

1. Az utasításnak megfelelő számú bájt beolvasása és kiírása, decimális és hexadecimális alakban.

Az utasítás disassemblálása és mnemonikjának kiírása. Az operandusok kiírása, amely tartalmazza az adatok áramlásának irányát is, illetve egyes utasításoknál egyéb, az utasításra vonatkozó információkat. Ezek a kezdők bővebb tájékoztatására szolgálnak.

Az állapotregiszter bitjei feletti kijelzés mutatja, hogy az utasítás hogyan befolyásolja az adott bitet.

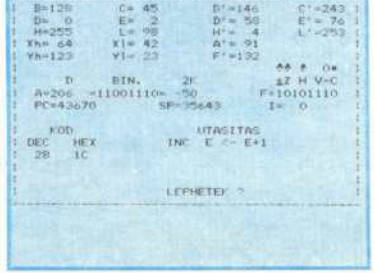
Az alkalmazott jelölések a következők:

* az utasítás a bitet nem változtatja meg,
↑ a változás a művelet eredményétől függően a bit értelmezése szerinti,
0 a bit értéke zérus lesz,
1 a bit értéke 1 lesz,
X a bit értéke előre nem határozható meg,
A ha A=(HL), akkor 1 lesz, egyébként zérus,
B ha B zérus lesz, akkor 1, egyébként zérus,
b ha BC zérus lesz, akkor 1, egyébként zérus,
I a bit a megszakítás flip-flop tartalmát fogja jelezni.

Ezután a képernyő alján feltűnik a „Léphetek?" kérdés. Ekkor a BRK kivételével bármely billentyűt megnyomva a 2. lépés következik.

2. Végrehajtódik a regiszterekkel a szimulált utasítás, és megjelennek az új regiszterértékek. A memóriába és a perifériaregiszterekbe író utasításokat a program nem hajtja végre, mert azok egyes címek esetén a szimulátor további működését lehetetlenné tennék. Ezeknél az utasításoknál a kiírandó értéket és a címet fel kell jegyezni, hogy valamely, később erről a címről olvasó utasítás szimulálásánál a helyes értéket beírhassuk. Az olvasóutasítások megkérdezik, hogy beolvasható-e az adat az adott címről. Ekkor „N"-nel válaszolva beírhatjuk a helyes értéket. Bármely más billentyűt megnyomva a program ténylegesen beolvassa a memóriából az adatot. Ezzel lezajlott egy utasítás szimulációja, és a „* Léphetek?" kérdés következik.

Az *ábra* a program egyik lehetséges képernyőjét mutatja egy utasítás beolvasása után (2. lépés előtti állapot).



Az SP regiszter kezelése

A program a verem szimulálására egy 200 bájtos területet tart fenn, egy belső mutatóval. Ebben tárolja a szimulált program veremadatait, függetlenül az SP regiszter aktuális értékétől. Ezért ha a szimuláció során mi átírjuk az SP értékét, vagy mert az SP regiszter direkt értékadás miatt megváltozik, tehát ha a verem tartalma is megváltozik, akkor a szimulátorral közölni kell az új veremtartalmat (ha van). Az új tartalom megadása előtt a régit kérésre kiírja a program, egyébként az elvész. Ha az új verem üres, N-nel kell válaszolni, mert az SP-változással a belső verem automatikusan üresnek indul. A programok többségénél erre nincs szükség, mert általában nem szokás több veremmel dolgozni, de a lehetőség nem zárható ki – és a szűkös memóriaméret miatt ez a megoldás látszott célszerűnek.

A Primo BASIC-interpretere mindössze egyszer használ ilyen utasítást, és akkor is üres az új verem, tehát semmit sem kell beírni.

A szimuláció menetébe történő beavatkozásnál a verem tartalma tetszés szerint módosítható, és ekkor az SP regiszter és vele együtt a belső mutató értéke automatikusan követi a változást. Az SP értékét ekkor nekünk nem szabad megváltoztatni! (Az eddigiekből következik, hogy az SP regiszter konkrét értékének a szimuláció szempontjából semmi jelentősége sincs.)

A Z80 processzor blokkos adatmozgató utasításainál a mozgatás részletes kiírását kérhetjük, ha a szimulátor által feltett kérdésre ilyenkor az I választ adjuk.

A szimuláció bárhol félbehagyható és máskor folytatható, ha az adott pillanatban érvényes regiszter- és veremtartalmakat feljegyezzük, majd később a folytatás kezdetén ezeket visszaírjuk a programnak.

A program az R tárfrissítő regiszter tartalmát nem írja ki, de a szimulációban természetesen ez is részt vesz. A program eredményes kipróbálásához hasznos kisegítő ismeretek találhatók a Primo szoftver- és hardverleírásában.

SZABADOS ISTVÁN

A cikkben szereplő program a szerkesztőségen keresztül klublemez formájában az érdeklődők részére hozzáférhető.

Integrált szoftver

Adatbázis II. Írjuk át saját gépünkre!

A múlt hónapban közöltük a programcsomag első tagjának, egy adatbázist kezelő programnak a használati utasítását és programlistáját. Mivel olyan programcsomagot ismertetünk, amelyet *minden olvasónk megvalósíthat a saját gépén*, ezért a programok BASIC nyelven íródtak, és mindegyikhez csatoljuk az átíráshoz szükséges tudnivalókat.

Az első részben sorra vettük azokat az utasításokat, amelyek az összes programban szerepelnek, és a DRAGON-BASIC specialitásai. Most azok következnek, amelyek itt ezeken felül előfordulnak. (A sorozat további részeiben ezt nem ismételjük!) Tehát az alábbi sorhivatkozások az előző számunkban közölt listára vonatkoznak. A folyamatot a program végrehajtási sorrendjében tárgyaljuk.

A 10-es sor helyfoglaló, képernyőszínező utasításai feleslegesek. Az 1030-as sorban a folytatásban szintén. Itt a MEM utasítás a BASIC fordító által szabadon hagyott memória méretét adja meg. Itt és máshol is a képernyőnek egy adott helyére kiíratott utasítás a képernyő méretétől függő értékre módosítandó. A DRAGON 16 sort és soronként 32 karaktert használ. Az 1050-es sor grafikus képmemóriát törlő utasítása — ahol létezik — a megfelelővel helyettesítendő.

A 60-as sor első, helyfoglaló utasítása csak ott kell, ahol a karaktersorozatoknak külön — az előre beállítottnál nagyobb — helyet lehet biztosítani. A harmadik utasításban levő szám helyett a szabad memória értékét írjuk be, természetesen figyelembe véve a program által elfoglalt helyet is. A STRING függvények elhagyhatók, vagy más díszítő elemmel helyettesíthetők.

A 200-as sorban a nyomtatóillesztő fajtájának megválaszthatósága a DRAGON specialitása, elhagyandó.

A 7110—9999-es sorokban előkészítjük az adatok kazettás egységre írását. Ehelyett a gépünknek megfelelő megoldást kell ideírni. A folytatás a 700-as sorban fájlnyitás, írás a fájlba, ami szintén gépfüggő, akárcsak a 710-es sorban a fájl lezárása.

A 660-as és 520-as sorban a LINE INPUT utasítást az INPUT utasítás helyettesíti, ha nem kívánunk vesszőt vagy más tiltott karaktert beadni. Egyébként itt a gépünknek megfelelő megoldásra van szükség. Fontos, hogy a DRAGON-BASIC az adat be- és kivitelnél 255 karaktert képes egy sorként kezelni.

Az 550-es és 950-es sorban az INSTR utasítás helyett a legtöbb gépen olyan szubrutint kell írnunk, amelyik a harmadiknak megadott karaktersorozatból kikeresi a második sorozatot az elsőnek megadott számú karaktertől kezdve. (Esetünkben S$-ból keresi T$(X)-et az első karaktertől kezdve.) Egy ilyen, általában használható szubrutin látható az *1. listában*. Itt A$-ból kell B$-t kikeresni C karaktertől kezdve. Az eredmény D. Amennyiben nem talál ilyet, D=0.

A 600-as sor fájlnyomtatás. Ez szintén olyan utasításcsoport, amit gépünkön valószínűleg másként kell megvalósítani.

A 8200—8250-es sorokban az ékezetes karaktereket állítjuk elő. A használt nyomtató (NEC, PR103A típus) „ismerte" a svéd és német karaktereket, így az ő, ő, ű, í kivételével minden karaktert írt, módosítás nélkül, az á helyett az Angström-jelet használva. A DRAGON billentyűzetén viszont nincsenek ékezetes karakterek, így ezeket a szövegben nem vagy csak ritkán előforduló kkal helyettesítettük. A nyomtatók sokféle lehet, tehát ez a rész sok esetben újrairandó (lásd a *táblázatot*).

A program átírásához itthon kevéssé ismert gépre szívesen adunk segítséget, és örömmel közlünk igényes megoldásokat.

Mivel a SZÖVEGSZERKESZTŐ program igen terjedelmes, most közöljük; *(2. lista)*.

DR. SIMONYI ENDRE

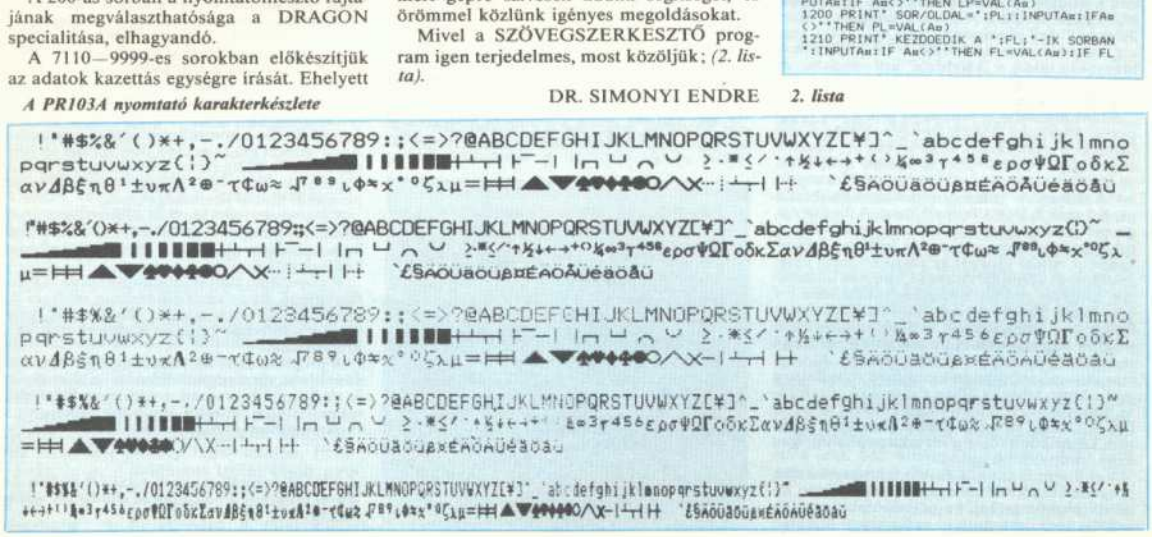
```
5000 E=LEN(B$)
5010 FOR I=C TO LEN(A$)-E
5020 IF MID$(A$,I,E)<>B$ THEN5040
5030 D=I:I=LEN(A$)-E
5040 NEXT I
```

1. lista

```
1 GOTO9000
10 CLEAR200:CLS2:PRINT@195,'    SZOEVEGSZERKESZTOE    ';:M=MEM
20 IF M>32000 THEN CLEAR30000:T=200 ELSE CLEAR 15000:T=100
30 IP=2
70 BF=T+150:FOR X=1 TO 2222:NEXT
80 DIMT$(T+2):OE=1:LM=10:LN=60:PL=54:FL=5:PP=0:S$=" "+STRING$(30,61):LP=66:BB$='A BLOKK ELEJE':BE$=' A BLOKK VEEGE':BT$='JELZOE'
500 CLS:POKE329,0:FS=0:FORX=0TO T:IF T$(X)=""THEN520ELSEFS=FS+LEN(T$(X))+4:NEXTX
520 FS=BF-FS:PRINT@384,S$;:PRINTUSING" ###,### ";FS;:PRINT'UERES HELYEK A LAP BALOLDALAAN':PRINT'AZ EERVEENYES ADATAALLOMAANYNEEV':DS=:PRINTe41,'SZOEVEGSZERKESZTOE':PRINTSS=
570 PRINT:PRINT' T - TIPUS',:PRINT'F - FORMAATUM':PRINT' B-BETOELTEES',:PRINT' K-KITOERLESES',:PRINT' S-SZERKESZTEES',:PRINT'G - OLDBALIS SZERKESZTEES':PRINT' E-ELTAAROLAAS',:PRINT'V-VISSZATOELTEES'
580 PRINT' M-MOZGATAAS',:PRINT'N-NYOMTATAAS':PRINT' U-UJRA'PRINT
650 PRINTTAB(9)'MIT VAALASZTO';:Q$=INKEY$:GOSUB680:IF Q$='F'THEN1000ELSEIF Q$='T'THEN2000ELSEIF Q$='M'THEN3420ELSEIF Q$='S'THEN4000ELSEIF Q$='E'THEN5000ELSEIF Q$='V'THEN6000ELSEIF Q$='N'THEN7000ELSEIF Q$='U'THEN8000
660 IF Q$='B'THEN3180ELSEIF Q$='K'THEN3040ELSEIF Q$='G'THEN4270
670 GOTO500
680 Q$=INKEY$:IFQ$=""THEN680ELSEO=VAL(Q$):RETURN
1000 CLS:PRINTe41,'OLDAL ELLENOERZEES':PRINTSS=:OO=" (O=K:,1=BE) :":PRINT' SZUENET';OO$; PP;:INPUTA$:IF A$<>""THEN PP=VAL(A$)
1040 PRINT' TOERDELVE';OO=:OE;:INPUTA$:IF A$<>""THEN OE=VAL(A$)
1060 PRINT' VEEDVE';OO$;JY;:INPUTA$:IF A$<>""THEN JY=VAL(A$)
1080 PRINT' KETTOES HELYEK';OO=:OS;:INPUTA$:IF A$<>""THEN OS=VAL(A$)
1120 PRINT'BALOLDALI MARGOO HELVE';LM;:INPUTA$:IF A$<>""THEN LM=VAL(A$)
1140 PRINT' SORHOSSZ';LN;:INPUTA$:IF A$<>""THEN LN=VAL(A$)
1160 PRINT' /o BEKEZDEES HELYE';PI;:INPUTA$:IF A$<>""THEN PI=VAL(A$)
1180 PRINT' OLDALHOSSZ SOROKBAN=';LP;:INPUTA$:IF A$<>""THEN LP=VAL(A$)
1200 PRINT' SOR/OLDAL=';PL;:INPUTA$:IFA$<>""THEN PL=VAL(A$)
1210 PRINT' KEZDOEDIK A ';FL;'-IK SORBAN':INPUTA$:IF A$<>""THEN FL=VAL(A$):IF FL
```

2. lista

A PR103A nyomtató karakterkészlete



```
=0THEN FL=1
1230 PRINT' OLDALSZAAM=';PG;:INPUT A¤:IF
 A¤<>''THEN PG=VAL(A¤)
1250 PRINT'FEJLEEC= ';HD¤;' ';:LINEINPUT
 A¤:IFA¤<>''THEN HD¤=A¤
1270 GOTO500
2000 CLS:PRINT:PRINT£42,'SZOEVEGBEADAAS'
:PRINTSS¤:PRINT£73,'PARANCSOK:';:PRINT£9
7,'/F   UJ OLDAL':PRINT' /aN   ASCII-KOO
DOK':PRINT' /c   POZICIONAALAAS':PRINT'
/i   MARGOO BEAALLITAAS'
2060 PRINT' /p   UJ BEKEZDEES':PRINT' /m
N  UGORJ AZ N#-IK SORRA':PRINT' /tN  TAB
ULAATOR POZICIOO':PRINT' /l   UJ SOR':PR
INT' /q   UJRA':PRINT'  e   A BEADOTT SZ
OEVEG SZERKESZTEESE';PRINTSS¤:INPUT'
reegi, VAGY uj';A¤
2070 IF A¤=uj' THEN PRINT' KEESZPRINT (I
/N)';:INPUT I¤:IF LEFT¤(I¤,1)='I' OR LEF
T¤(I¤,1)='i' THEN 2120 ELSE 500
2120 IF A¤='uj'THEN FOR X=0 TO T:T¤(X)='
':NEXTX:N=0:DS¤='':AL¤='':GOTO2160
2130 IF A¤=reegi'THEN2140ELSE2000
2140 FOR N=0TO T:IF T¤(N)<>''THEN NEXT N
2150 IF N>0 THENPRINT:PRINTT¤(N-1):PRINT
 '#';N:IFN > T THEN500
2160 LINEINPUT' SZOEVEG>';T¤
2170 IF LEFT¤(T¤,2)='/q'OR T¤='q' THEN 5
00
2177 IF T¤='e'THEN N=N-1:E¤='BE':GOTO409
0
2180 IF LEFT¤(T¤,1)<>'/'THENT¤='   '+T¤
2190 T¤(N)=T¤:N=N+1:IFN > T THEN500
2200 PRINT:GOTO2160
3040 CLS:PRINT:PRINT' BETOLDAAS':PRINT
3050 PRINT:INPUT'  # SORT, q, VAGY '';A
¤
3060 IF A¤='q'THEN 500
3070 IF A¤='0'THEN N=0:GOTO3100
3080 NN=VAL(A¤):IF NN > 0 AND NN<= T THE
N N= NN:GOTO3100
3090 GOSUB3290
3100 CLS:IF N>0THEN PRINT:PRINTN-1;T¤(N-
1)
3110 PRINT:PRINTN;T¤(N):PRINT:PRINT' AZ
';N;'-IK SOR ELEE':INPUTA¤:IF A¤<>'y'THE
N3050
3130 FORX=T-1TO N STEP-1:T¤(X+1)=T¤(X):N
EXTX:T¤(N)=' ':LINEINPUT' SZOEVEG>';T¤:I
F LEFT¤(T¤,1)<>'/'THEN T¤='   '+T¤
3160 T¤(N)=T¤:GOTO3040
3180 CLS:PRINT:PRINT' TOERLEES':PRINT
3190 PRINT:INPUT' # SORT, q, VAGY '';A¤
:IF A¤='q'THEN500
3210 IF A¤='0'THEN N=0:GOTO3240
3220 NN=VAL(A¤):IF NN > 0 AND NN <= T TH
EN N= NN:GOTO3240
3230 GOSUB3290
3240 CLS:PRINT:PRINTN;T¤(N)
3250 PRINT:INPUT' TOERLI EZT A SORT';A¤:
IF LEFT¤(A¤,1)<>'i'THEN3190
3270 FORX=N TO T-1:T¤(X)=T¤(X+1):NEXTX:T
¤(T)='':GOTO3190
3290 CLS:PRINT:PRINTTAB(9)'SOR KERESEES'
:PRINTSS¤:IFN > T THENN = 0
3300 PRINT:PRINTN;T¤(N):FORX=1TO200:NEXT
X:FORX=338TO345:POKE(X),255:NEXTX
3320 A¤=INKEY¤:IF A¤=''THEN3320
3330 IF A¤=CHR¤(94)AND N>0THEN N=N-1
3340 IF A¤=CHR¤(95)AND N>7THEN N=N-8
3350 IF A¤='b'THEN BB = N:PRINT:PRINTBB¤
3360 IF A¤='e'THEN BE = N:PRINT:PRINTBE¤
3370 IF A¤='t'THEN BT = N:PRINT:PRINTBT¤
3380 IF A¤=CHR¤(10)AND N < T THEN N=N+1
3390 IF A¤=CHR¤(91)AND N < T-8 THEN N=N+
8
3400 IF A¤=CHR¤(13)THEN RETURN
3410 GOTO3300
3420 GOSUB3660:PRINT' BEADANDOO':BB=0:BE
=0:BT=0:PRINT' <B>   ';BB¤,' <E>   ';BE¤,'
 <T>   ';BT¤:INPUT' KEESZ';A¤:IF A¤='u'TH
EN500
3460 GOSUB3290:GOSUB3660:PRINTBB¤;'=';BB
:PRINTBE¤;'=',BE:PRINTBT¤;'=',BT:IF BB >
 BE THEN 3670
3490 IF BB <= BT AND BT <= BE THEN3670
3500 DF=BE - BB:IF BT < BB THEN3520ELSE3
590
3520 FOR X=0 TO DF:PRINT£193,'SOR';BB+X;
'MOZGATAAS':HA¤=T¤(BB+X):FOR Y= BB+X TO
BT+X+1 STEP-1:T¤(Y)=T¤(Y-1):NEXTY:T¤(Y)=
HA¤:NEXTX:GOTO500
3590 BT = BT-1:FOR X= 0 TO DF:PRINT£193,
'SOR';BE-X;'MOZGATAAS':HA¤=T¤(BE-X):FOR
Y= BE - X TO BT - X-1:T¤(Y)=T¤(Y+1):NEXT
Y:T¤(Y)=HA¤:NEXTX:GOTO500
3660 CLS:PRINT:PRINT' BLOKK MOZGATAAS':P
RINTSS¤:RETURN
3670 PRINT' TILTOTT MOZGATAAS':FORX=1TO2
000:NEXTX:GOTO3420
4000 CLS:PRINT£42,'SZERKESZTEES':PRINTSS
¤
4040 PRINT:INPUT' # SORT, q, VAGY '';A¤
:IF A¤='u'THEN500
4060 IF A¤='0'THEN N=0:GOTO4090
4070 NN=VAL(A¤):IF NN>0 AND NN <= T THEN
 N = NN:GOTO4090
4080 GOSUB3290
4090 LL=0:T¤=T¤(N)
4100 CLS:PRINT£483,'<I>-BETOLDAAS <C>-CS
ERE <D>-TOERLEES':
4110 PRINT£44,'SZERKESZTEES':PRINTSS¤:PR
INT£96,MID¤(T¤,1,LL);:PRINT£96+LL,CHR¤(1
59);:PRINT£97+LL,MID¤(T¤,LL+1):I¤=INKEY¤
4120 IFI¤=CHR¤(8) THENIFLL>0THENLL=LL-1:
GOTO4110
4130 IFI¤=CHR¤(9) THENIFLL<LEN(T¤)THENLL
=LL+1:GOTO4110
4140 IFI¤=CHR¤(94) THENIFLL=>32THENLL=LL
-32:GOTO4110
4150 IFI¤=CHR¤(10) THENIFLL+32<=LEN(T¤)T
HENLL=LL+32:GOTO4110
4160 I¤=INKEY¤:IFI¤=''THEN4160
4170 IFI¤=CHR¤(8)ORI¤=CHR¤(9)ORI¤=CHR¤(9
4)ORI¤=CHR¤(10)THEN4120
4180 IFI¤<>'d'THEN4210
4190 PRINT£44,'TOERLEES':I¤=INKEY¤:IFI¤=
''THEN4190ELSEIFI¤<>'d'THEN4100
4200 SOUND160,1:IFLEN(T¤) >LL THENT¤=LEF
T¤(T¤,LL)+RIGHT¤(T¤,LEN(T¤)-LL-1):PRINT£
97+LL,MID¤(T¤,LL+1):SOUND140,1:GOTO4190
4210 IFI¤<>'i'THEN4230
4220 PRINT£44,'BETOLDAAS':PRINT£97+LL,ST
RING¤(100,32):PRINT£192+LL,MID¤(T¤,LL+1)
:PRINT£LL+96,'';:LINEINPUTI¤:T¤=LEFT¤(T¤
,LL)+I¤+RIGHT¤(T¤,LEN(T¤)-LL):LL=LL+LEN(
I¤):GOTO4100
4230 IFI¤<>'c'THEN4250
4240 PRINT£44,'CSERE':PRINT£LL+96,'';:LI
NEINPUTI¤:IFLEN(I¤)> LEN(T¤)-LL THENT¤=L
EFT¤(T¤,LL)+I¤ELSET¤=LEFT¤(T¤,LL)+I¤+RIG
HT¤(T¤,LEN(T¤)-LL-LEN(I¤)):LL=LL+LEN(I¤)
:GOTO4100
4250 IFI¤=CHR¤(13)THENT¤(N)=T¤:IF E¤='BE
'THEN E¤='':CLS:GOTO2140 ELSE 4040
4260 GOTO4110
4270 CLS:PRINT:PRINTTAB(5)'GLOBALIS SZER
KESZTEES':PRINTSS¤:PRINT:LINEINPUT' q, V
AGY AALLOMAANYKERESEES> ';S¤:IF S¤='q'TH
EN500
4300 PRINT:LINEINPUT' BEADANDOO AZ UJ>';
R¤
4310 FORX= 0 TO T
4315 P=1
4320 P=INSTR(P,T¤(X),S¤):IF T¤(X)=''THEN
 X=T:NEXTX:GOTO500
4340 IF P=0 THEN NEXTX
4350 CLS:PRINT:PRINT' POSICIO';P;',';:PR
INT' #';X;';'HIVATKOZAAS';'VOLT':PRINTT¤
(X):T¤=T¤(X):PRINT:PRINT TAB(12)'JOEN':T
¤=LEFT¤(T¤,P-1)+R¤+RIGHT¤(T¤,LEN(T¤)-P+1
-LEN(S¤)):PRINTT¤:INPUT'KEESZ  (i, n OR
q)';A¤
4410 IF LEFT¤(A¤,1)='q'THEN500
4420 IF LEFT¤(A¤,1)='n'THEN P=P+1:GOTO43
20
4425 IFR¤='f'THENN=X:X=T:NEXTX:GOTO4090
4430 T¤(X)=T¤:GOTO4315
5000 CLS:PRINT:PRINT' AALLOMAANYELTAAROL
AAS':POKE329,255:PRINTSS¤:PRINT' EERVEEN
YES AALLOMAANYNEEV=':DS¤:PRINT:INPUT' AA
LLOMAANYNEEV>';DF¤:IFLEFT¤(DF¤,1)='U'THE
N500ELSE GOSUB5100
5020 IF DF¤<>''THEN DS¤=DF¤
5030 PRINT:PRINTTAB(7)'ELTAAROLAAS';DS¤
5040 OPEN'O',-1,DS¤:PRINT#-1,LM,LN,PL,FL
,PG,OE,HD¤,PI,JY
5050 FOR X=0 TO T
5060 IF T¤(X)=''THEN X=T:GOTO5080
5070 PRINT#-1,T¤(X)
5080 NEXTX
5090 CLOSE-1:GOTO500
5100 FOR X=1 TO 20:PRINT£320,' ':SOUND20
0,1:PRINT£325,'A MOTOR BEKAPCSOLVA':SOUN
D200,1:NEXT X:MOTORON:FOR X=1 TO 4400:NE
XTX:RETURN
6000 CLS:PRINT:PRINT' AALLOMAANYTOELTEES
':POKE329,255:PRINTSS¤
6020 INPUT' AALLOMAANYNEEV>';DS¤:DS=0:LI
=0:IFDS¤='U'THEN500
6040 FORX=0 TO T:T¤(X)='':NEXTX:OPEN'I',
-1,DS¤:INPUT#-1,LM,LN,PL,FL,PG,OE,HD¤,PI
,JY
6070 FOR X=0TO T:LINEINPUT#-1,T¤(X):IF E
OF(-1)THEN X=T
6100 NEXTX
6110 CLOSE-1:IF AL¤='BE'THEN7000ELSE500
7000 CLS:PRINT:PRINT' SZOEVEGNYOMTATAAS'
:PRINTSS¤:PI=PG:PS=0:SI=0:TAB=LM:N=0:PRI
NT'AZ OLDAL FORMAATUMA RENDBEN ?':GOSUB6
80:IF O¤<>'i'THEN1000ELSE INPUT' KEZDEES
HELYE';N
7010 IF IP=2THEN PRINT'FORMAALJA?':GOSUB
680:IF O¤='n'THEN IP=0ELSE IP=1
7020 IF SP<>1THEN  SP=1:PRINT' SOROS, VA
GY PAARHUZAMOS NYOMTATOO       (s OR p)?'
:GOSUB680:IF O¤='s'THENPOKE&H3FF,1 ELSEP
OKE&H3FF,0
7040 CLS:PRINT:PRINT' NYOMTATAAS':PRINTS
S¤
7050 OPEN'O',-2,'PRINTER':GOSUB7720
7080 A¤=T¤(N)
7090 IF LEFT¤(A¤,1)=' 'THENA¤=RIGHT¤(A¤,
LEN(A¤)-1):GOTO7090
7100 N=N+1:IFN > T THEN7780ELSE IF CL>=
PL ANDLEFT¤(A¤,2)<>'/F'THEN A¤='   '+A¤:G
OSUB7670
7120 IF LEFT¤(A¤,1)<>'/'THEN 7430ELSE IF
 LEFT¤(A¤,2)='/d'THENDS¤=RIGHT¤(A¤,LEN(A
¤)-2):CLOSE:AL¤='ON':GOTO6000
7130 IFLEFT¤(A¤,2)='/k' THENPRINTA¤:LINE
INPUTA¤:GOTO7120
7140 IFLEFT¤(A¤,2)<>'/a'THEN7220ELSE PC¤
=MID¤(A¤,3,3):PC=VAL(PC¤)
7160 IFPC>255THENPRINT£228,' ASCII-KOD':
GOTO7080
7170 PRINT#-2,CHR¤(PC);:PC¤=STR¤(PC):X=L
EN(PC¤)-1:A¤=RIGHT¤(A¤,LEN(A¤)-X-2):IF A
¤<>''THEN7140
7210 PRINT#-2,' ':CL=CL+1:GOTO7080
7220 IF LEFT¤(A¤,2)='/F'THENGOSUB7670:GO
TO7120
7230 IFLEFT¤(A¤,2)='/c'THEN GOTO 7810
7240 IFLEFT¤(A¤,2)<>'/i'THEN7290
7250 IF SI=1 THEN7270
7260 SI=1:LM=LM+5:LN=LN-10:GOTO7280
7270 SI=0:LM=LM-5:LN=LN+10
7280 A¤=RIGHT¤(A¤,LEN(A¤)-2):TAB=LM
7290 IF LEFT¤(A¤,2)<>'/p'THEN7320
7300 PRINT#-2,' ':A¤=RIGHT¤(A¤,LEN(A¤)-2
):PS=1:TAB=TAB+PI:LN=LN-PI:CL=CL+1:IF DS
=1THENPRINT#-2,' ':CL=CL+1
7310 GOTO7120
7320 IF LEFT¤(A¤,2)<>'/s'THEN7360
7330 X¤=MID¤(A¤,3,2):X=VAL(X¤):FOR Y=1TO
X:PRINT#-2,' ':CL= CL+1:NEXTY:X¤=STR¤(X
):X=LEN(X¤)-1:A¤=RIGHT¤(A¤,LEN(A¤)-X-2):
GOTO7120
7360 IF LEFT¤(A¤,2)<>'/t'THEN7410
7370 X¤=MID¤(A¤,3,2):TAB=VAL(X¤):X¤=STR¤
(TAB):X=LEN(X¤)-1:IF TAB<0 THEN X=X+1
7390 A¤=RIGHT¤(A¤,LEN(A¤)-X-2):TAB=TAB+L
M:T¤=A¤:L=LEN(A¤):GOTO7430
7410 IF LEFT¤(A¤,2)<>'/l'THEN7430ELSE A¤
=RIGHT¤(A¤,LEN(A¤)-2)
7430 IF LEN(A¤)>LN THEN7480
7440 ZZ¤=A¤:GOSUB8200:A¤=ZZ¤:IF T¤(N)=''
THEN PRINT#-2,TAB(TAB)A¤:GOTO7780
7450 IF LEFT¤(T¤(N),1)='/'THEN T¤=A¤:L=L
EN(A¤):GOTO7550
7460 IF LEN(A¤)+LEN(T¤(N))>250THENGOSUB8
040
7470 A¤=A¤+T¤(N):N=N+1:IFN > T THEN7780E
LSE7430
7480 IF LEFT¤(A¤,1)=' 'THEN A¤=RIGHT¤(A¤
,LEN(A¤)-1):GOTO7480
7490 L=1:LL=L
7500 LL=INSTR(LL,A¤,' '):IF LL= 0AND LEN
(A¤)<=LN THEN L=LEN(A¤):GOTO7540
7520 IF LL= 0AND LEN(A¤)> LN THEN7540
7530 IF LL <= LN+1 THEN L = LL:LL=LL+1:G
OTO7500
7540 T¤=LEFT¤(A¤,L)
7550 IF LEFT¤(T¤,1)=' 'THEN T¤=RIGHT¤(T¤
,LEN(T¤)-1):GOTO7550
7560 IF CL>= PL THEN A¤='   '+A¤:GOSUB767
0
7570 IF LI =1 THENPRINT#-2,N-1;
7580 IF JY=1 THENGOSUB8110
7590 ZZ¤=T¤:GOSUB8200:T¤=ZZ¤:PRINT#-2,TA
B(TAB);T¤:CL= CL+1:IF DS=1THENPRINT#-2,'
 ':CL=CL+1
7600 PRINT£167,' #';CL;'-IK SOR';'NYOMTA
TVA':PRINT£288,STRING¤(99,32):PRINT£257,
T¤:TAB = LM:IFPS=1 THENLN=LN+PI:PS=0
7610 QT¤=INKEY¤:IF QT¤='Q'OR QT¤='q'THEN
7780
7630 A¤=RIGHT¤(A¤,LEN(A¤)-L):IF LEN(A¤)>
 LN THEN 7480
7650 IF A¤=''THEN7080
7660 GOTO7440
7670 IF IP=1THENPRINT#-2,CHR¤(12);:GOTO7
690
7680 FORF = CL+FL TOLP:PRINT#-2,' ':NEXT
F
7690 A¤=RIGHT¤(A¤,LEN(A¤)-2)
7700 IF PP=1THEN SOUND120,18:PRINT£228,'
BEADANDOO <ENTER> A FOLYTATAASHOZ';:GOSU
B680:PRINT£228,' '
7720 IFFL>1THENFOR X=1TO FL:PRINT#-2,' '
:NEXTX
7730 CL=1:IF PG=0 THEN RETURN
7740 IF PG=1THEN PG=PG+1:RETURN
7745 IF SI=1THEN LM=LM-5:LN=LN+5
7750 ZZ¤=HD¤:GOSUB8200:HD¤=ZZ¤:PRINT#-2,
TAB(LM)HD¤;:PRINT#-2,TAB((TAB+LN)-7)'Old
al';PG:CL= CL+2:IF DS=1 THEN PRINT#-2,'
':CL=CL+1
7755 IF SI=1THEN LM=LM+5:LN=LN-5
7770 PG=PG+1:PRINT#-2,' ':RETURN
7780 IF OE=1AND IP=1THENPRINT#-2,CHR¤(12
);
7785 IF OE=1ANDIP=0THENFORX=CL+FL TO LP-
1:PRINT#-2,' ':NEXTX
7790 IF PS=1 THEN LN=LN+PI:PS=0
7800 CLOSE:PG=P1:X=0:N=0:GOTO500
7810 A¤=RIGHT¤(A¤,LEN(A¤)-2):ZZ¤=A¤:GOSU
B8200:A¤=ZZ¤:PRINT#-2,TAB((LN-LEN(A¤))/2
+LM)A¤:CL=CL+1:IF DS=1THENPRINT#-2,' ':C
L=CL+1
7820 GOTO7080
8000 CLS:PRINT:PRINT' AALLOMAANY ELTAARO
LVA?':GOSUB680:IF O¤<>'i'THEN500ELSE CLO
SE:POKE329,255:STOP
8030 GOTO500
8040 PRINT£228,'#';N;' AZONOSITO':FOR X
= T TO N+1STEP-1:T¤(X)=T¤(X-1):NEXTX:X=L
EN(T¤(N))/2:X=INT(X):T¤(N+1)=RIGHT¤(T¤(N
),LEN(T¤(N))-X):T¤(N)=LEFT¤(T¤(N),X):RET
URN
8110 IF T¤=A¤ AND LEN(A¤) < LN-7 OR LEN(
T¤) < LN-12 THEN RETURN
8120 IF RIGHT¤(T¤,1)=' 'THEN T¤=LEFT¤(T¤
,LEN(T¤)-1):GOTO8120
8130 P=RND(LN)
8140 P=INSTR(P,T¤,' '):IF LEN(T¤)>=LN TH
ENRETURN
8160 IF P=0 THEN P=1:GOTO 8140
8170 T¤=LEFT¤(T¤,P)+' '+RIGHT¤(T¤,LEN(T¤
)-P):P=P+3:GOTO8140
8200 ZY¤='':FORX=1TO LEN(ZZ¤):ZX¤=MID¤(Z
Z¤,X,1):IF (ZX¤<>'%')AND(ZX¤<>'¤')AND(ZX
¤<>'&')AND(ZX¤<>'#') THEN8250
8210 IF ZX¤='#' THEN ZX¤=CHR¤(125):GOTO8
250
8220 IF ZX¤='%' THEN ZX¤=CHR¤(124):GOTO8
250
8230 IF ZX¤='¤' THEN ZX¤=CHR¤(96):GOTO82
50
8240 ZX¤=CHR¤(126)
8250 ZY¤=ZY¤+ZX¤:NEXT X:ZZ¤=ZY¤:RETURN
9000 CLS0:PCLEAR1:GOTO10
9999 MOTORON:FOR X=1 TO 8000:NEXT:FORX=1
 TO2:CSAVE'SZERKESZ':MOTORON:FORZ=1TO999
:NEXTZ,X:MOTOROFF
```

...persze most is, de mondandónk megint csak lazán kapcsolódik olvasmányainkhoz. Az ötlet és az indítás ismét kedvenc könyvünkből származik (Wirth: Algoritmusok + Adatstruktúrák = Programok). Az úgynevezett „visszalépéses algoritmusokról" lesz szó (visszalépés angolul: backtracking). Az iskola (ideérve az egyetemeket is) általában olyan feladatokat ad, amelyeknek valamilyen deduktív, *levezetéses* megoldásuk van, és az eredményt valamilyen jól meghatározott szabály, képlet alkalmazásával kaphatjuk meg. Emiatt sokunkban eleinte az a téves elképzelés alakul ki, hogy

— minden probléma megoldható deduktív, levezetéses úton,

— a feladat minden esetben csupán a levezetés (a megoldóképlet) megtalálásának kérdése.

Az élet azonban tele van levezetéssel elvileg vagy gyakorlatilag *nem* megoldható problémával. Az ún. „általános problémamegoldás" kérdése a számítógépes programozási törekvések egyik érdekes területe. Olyan algoritmusok megtalálása a cél, amelyek a problémát nem meghatározott kiszámítási szabályok alapján, hanem pl. *módszeres próbálgatással* oldják meg. A feladatok megoldása módszeres próbálgatással természetesen számítógép nélkül is megkísérelhető, csak hát az ember figyelme véges: előbb-utóbb hibázik, és nem találja meg a megoldást. Gyakran — még az igen terjedelmes keresési feladatoknál is — maga a keresés algoritmusa az ember számára még áttekinthető, és megfelelő nyelven programba foglalható. Ha ezt a programot jól írtuk meg, akkor a számítógép — mivel figyelme nem lankad, nem fárad el —, ha kell, akár több napi munkával, de megtalálja a megoldás(oka)t, vagy — ami ezzel egyenértékű — kimutatja, hogy nincs megoldás. A módszeres kereséssel megoldható problémák osztályát általában valamilyen egyszerű, játékos példán szokták bemutatni. Ilyen „a huszár útja a táblán" feladat:

Egy $n \times n$ méretű sakktábla valamelyik mezőjére egy huszárt helyezünk. A feladat annak az útnak (azoknak az utaknak) a megtalálása, amelyet bejárva a huszár mindegyik mezőt legalább egyszer — de nem többször (!) — érinti. Néhány kísérlet után felépítjük a módszeres próbálgatás algoritmusát, ezt folyamatábrán ábrázoljuk, majd bemutatjuk az algoritmusnak megfelelő programot.

Könnyen belátható, hogy 3×3 méretű sakktáblán nincs megoldás. Ha a huszár középről indulna, akkor nem lenne hová

(25)	2	13	8	23
12	7	24	3	14
(1)	18	15	22	9
6	11	20	17	4
19	16	5	10	21

1. ábra

			2
	(1)		
		3	
(4)			

2. ábra

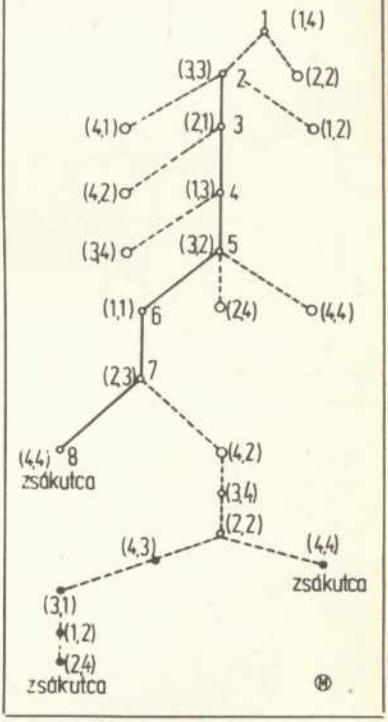
	4	7	(10)
8	(1)		5
3	6	9	
		2	

3. ábra

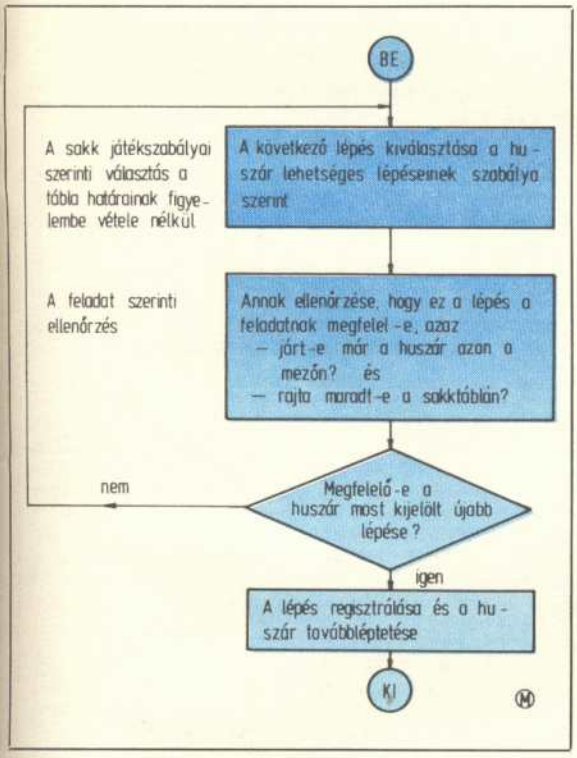
	8		1	
7				2
		♞		
6				3
	5		4	

4. ábra

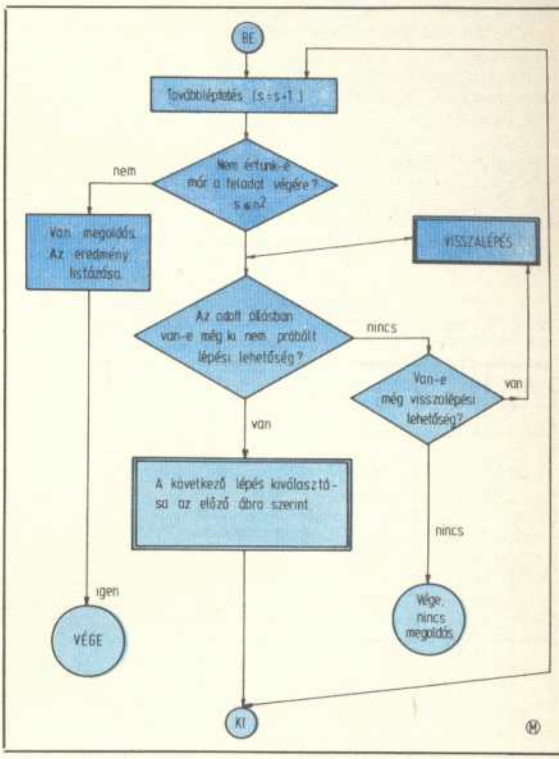
(1)			(8)
4	7	2	
		5	
6	3		



5. ábra



6. ábra



7. ábra

érkeznie: bármelyik irányba mozdulna, lelépne a tábláról. Akármelyik szélső mezőről kezd, sohasem juthat el a középső mezőre.

5 × 5 méretű táblán már van megoldás (1. ábra)

A mezőkben a lépés sorszámát tüntettük fel. A huszár a „bal középről" indult, és a jobb felső sarokban fejezte be útját, közben minden mezőt csak egyszer érintett. Kérdés: innen indulva csak ez az egy lehetséges megoldás képzelhető-e el?

Egy 4 × 4-es táblán a 2. ábra szerinti útvonallal gyorsan zsákutcába kerülünk.

Másképp elindulva már tíz lépésig jutunk el. (3. ábra)

Most eláruljuk: nincs megoldás a 4 × 4-es táblán.

De hogyan lehetünk biztosak felőle, hogy az összes lehetséges utat kipróbáltuk? Ez a kérdés elvezet a rendszeres, szisztematikus keresés problémájához. Induljunk ki onnan, hogy a huszár minden egyes mezőről legfőljebb nyolc másik mezőre léphet. (4. ábra)

Azért legfőljebb, mert nem léphet le a tábláról, és nem léphet olyan mezőre sem, amelyre már előzőleg rálépett. Szisztematikusan akkor járunk el, ha minden egyes állásnál az összes lehetséges (tehát 8) irányt kipróbáljuk.

Mi történik akkor, ha valamelyik irányban elindulva *zsákutcába* jutunk? *Visszalépünk* (innen az algoritmus neve!) az előző állásba, és onnan a soron következő irányt választjuk. Ha innen is minden irány zsákutcába vezet, akkor ismét egyet *visszalépünk*, és itt próbáljuk ki az újabb irányt (irányokat). Az ilyenfajta keresést ún. „keresőfával" is lehet írni. (5. ábra)

A fán felülről lefelé haladva világos, hogy ha a 7. lépés után nem a (4,4) koordinátájú mezőt választjuk, akkor a (4,2) irányban még egy jó darabig továbbmehettünk volna, de így sem juthatunk el a megoldásig. A fáról látható, hogy nincs más megoldás: *vissza kell lépnünk* egészen az 5. lépésig, és ott az első (1,1) választás helyett egy másik irányba — pl. a (2,4) irányba kellene haladni. Ha ez is zsákutcába vezetne (oda is vezet!), akkor itt még próbálkozhatunk a (4,4) iránnyal. Ha ez sem lenne jó (nem lesz jó!), akkor ismét *vissza kellene lépni* a negyedik állásig, és ott kísérletezni a (3,4) iránnyal.

Az egész folyamatot egy olyan próbálgatásos vagy kereső módszernek tekintjük,

19	28	35	26	21	6
36	(1)	20	7	34	25
29	18	27	24	5	22
12	15	2	31	8	33
17	30	13	10	23	4
14	11	16	3	32	9

8. ábra

```
 100 PRINT "************************"
 110 PRINT "   HUSZAR A TABLAN      "
 120 PRINT "************************"
 130 REM      *** BEALLITASOK ***
1000 PRINT " TABLAMERET ?"
1010 INPUT N
1020 IF N<3 THEN 1010
1030 IF N>8 THEN 1010
1032 REM AZ ELFOGADHATO LEGKISEBB
1033 REM TABLA 3X3-AS LEGN. 8X8-AS
1035 N2=N^2
1037 REM N2 A MEZOK SZAMA A TABLAN
1040 PRINT "HONNAN INDULJON ?"HUSZAR
1050 INPUT X0,Y0
1060 IF X0<1 THEN 1050
1070 IF X0>N THEN 1050
1080 IF Y0<1 THEN 1050
1090 IF Y0>N THEN 1050
1100 K=0
1102 REM K A MEGTETT LEPESEKPESEKET
1103 REM ES VISSZALEPESEK SZAMA
1105 REM A HUSZAR LEPESTABLAZATA:
1110 DIM C(8,2)
1120 C(1,1)=1
1130 C(1,2)=2
1140 C(2,1)=2
1150 C(2,2)=1
1160 C(3,1)=2
1170 C(3,2)=-1
1175 C(4,1)=1
1180 C(4,2)=-2
1190 C(5,1)=-1
1200 C(5,2)=-2
1210 C(6,1)=-2
1220 C(6,2)=-1
1230 C(7,1)=-2
1240 C(7,2)=1
1250 C(8,1)=-1
1260 C(8,2)=2
1300 REM A LEPESTABLAZAT VEGE
1400 REM " IRANY TABLAZAT ",MELY
1405 REM LEPESENKENT 8 IRANYT TARTALMAZ
1420 DIM B(64)
1430 REM A JO UTVONALON  MEGTETT
1432 REM LEPESEK SZAMA=N^2-1 (S=N2)
1600 X1=X0
1610 Y1=Y0
1700 S=1
1710 REM "S" AZ ELORE TETT LEPESEK
1715 REM SZAMA. A KIINDULAS S=1
1800 DIM A(8,8)
1801 REM A "SAKKTABLA" MAXIMALIS
1805 REM MERETEBEN
1815 A(X0,Y0)=1
1816 REM AZ ELSO LEPES FELJEGYZESE A
1818  REM TABLARA
1960 REM ********************************
1970 REM MOST MEGPROBALUNK ELORE LEPNI
2040 S=S+1
2050 K=K+1
2200 IF B(S)>7 THEN 9000
2205 REM UJABB LEPESI LEHETOSEG
2207 REM ITT NINCS: VISZALEPES=B(S)]]]
2300 REM * A LOVEGRAS SZABALYAI SZERINT
2310 REM 8-FELE LEPES GENERALASA  ****
2340 B(S)=B(S)+1
2345 REM B(S) 1-8 KOZOTTI ERTEKEIHEZ
2347 REM C(E)-N KERESZTUL EGY-EGY LEPES
2348 REM TARTOZIK
2350 X=X1+C(B(S),1)
2410 REM ** C A LOVEGRASOKAT, B PEDIG
2415 REM AZ ELLOL SORON KOVETKEZOT **
2420 Y=Y1+C(B(S),2)
2430 REM A C TOMB 1-ES OSZLOPA AZ X,
2440 REM A 2-ES AZ Y NOVEKMENYEIT TART
3300 REM A LEPES ELFOGADHATO
3305 REM VOLTANAK ELLENORZESE KOVETK.
3310 IF Y<1 THEN 2200
3320 IF X>N THEN 2200
3330 IF Y>N THEN 2200
3340 IF X<1 THEN 2200
3350 IF A(X,Y)<>0 THEN 2200
3360 REM AZAZ HA EZT A MEZOT MAR
3370 REM ERINTETTE A HUSZAR
3400 REM***  HA EGYIK KIZARO
3410 REM FELTETEL SEM TELJESULT  A LEPEST
3415 REM FELJEGYEZZUK
3420 REM AZAZ ERTUK A GEPI TABLARA
3425 X1=X
3426 Y1=Y
3427 A(X1,Y1)=S
3440 REM AZAZ TOVABBLEPTUNK
3450 IF S>=N2 THEN 11000
3460 REM AZAZ TALATUNK MEGOLDASTJJJ
3525 GOTO 2040
9000 REM ** VISZALEPTETES-RUTIN ****
9050 IF S<3 THEN 12000
9060 B(S)=0
9100 A(X1,Y1)=0
9110 REM A "TABLAI" TETT LEJEGYZEST
9120 REMTOROLJUK,MERT ZSAKUTCA VOLT
9200 S=S-1
9205 K=K+1
9300 REM *** MOST MEGKERESSUK A
9305 REM VISSZALEPES UTANI HELYET **
9310 X1=X1-C(B(S),1)
9320 Y1=Y1-C(B(S),2)
9340 REM A VISSZALEPES MIATT KIVONAS
9400 GOTO 2200
9410 REM VISSZALEPTUNK ES
```

amely lépésről lépésre a részfeladatok egy-egy fáját építi és vizsgálja (azaz elveti és lenyesi, ha zsákutcába jut). A legtöbb problémánál a fa igen gyorsan — többnyire exponenciálisan — terebélyesedik. Ennek megfelelően növekszik a keresési munka is. A bemutatott példa legfontosabb vonása, hogy mindig egy újabb lépéssel igyekszik a teljes megoldáshoz közelebb jutni, ezt fel is jegyzi, de előfordulhat, hogy egy korábbi lépést törölnie kell, mivel bebizonyosodik, hogy a lépés zsákutcába viszi. Ezt a műveletet nevezik visszalépéses haladásnak (*backtracking*-nek).

Az algoritmus *magja* a huszár következő lépésének kiválasztása. (6. ábra)

Most vegyük elő a huszár lehetséges lépéseinek generálását. Ha a huszár az (x, y) koordinátájú mezőn áll, akkor a következő lépés az alábbi mezőre visz:

x = x + c (i,1)
y = y + c (i,2)

ahol a c (i,z) kétdimenziós tömb (mátrix), mely a huszár lehetséges lépéseit tartalmazza (lásd a táblázatot).

A visszalépés esetén a következő:

x = x − c(i,1)
x = y − c(i,2)

Az i index a lehetséges lépések sorszáma. A c(i,1) értékek x növekményei, a c(i,2) értékek pedig y növekményei. Meg kell oldani az egy-egy állásnál kipróbált irányok nyilvántartását is. Ennek helye legyen a b

```
 9420 REM MEGFOGALUNK ELORE LEPNI
 9500 STOP
11000 REM ** AZ EREDMENY LISTAZASA **
11050 PRINT "*************************"
11060 PRINT "       AZ EREDMENY:      "
11200 FOR Y=N TO 1 STEP -1
11250 PRINT
11300 PRINT
11310 PRINT A(1,Y),A(2,Y),A(3,Y);A(4,Y);A(5,Y);A(6,Y)
11350 PRINT A(5,Y);A(6,Y)
11400 NEXT Y
11450 PRINT "A LEPESEK ES"
11460 PRINT " VISSZALEPESEK SZAMA K=";K;CPU-IDO=";TIM(0)
11470 PRINT " CPU-IDO=";TIM(0)
11550 PRINT "*************************"
11560 PRINT "       VEGE
      "
11570 PRINT "*************************"
11600 STOP
12000 PRINT "** NINCS MEGOLDAS, VEGE **"
12010 PRINT " K=",K," CPU-IDO=";TIM(0)F(0)
12100 STOP
```

Táblaméret	Kiindulás		A lépések száma (k)	CPU-idő* (sec)	Megoldás van/nincs
	x^0	y^0			
3	2	2	1	0,2—0,3	nincs
3	1	1	29	0,18—0,27	nincs
3	2	3	29	0,18—0,27	nincs
4	1	1	4445	7—8**	nincs
4	4	4	4445	7—8	nincs
4	2	2	3001	~5	nincs
5	1	5	56	~0,4	van
5	1	1	17654 (!)	~29 (!)	van
5	5	5	550	~1,2***	van
5	5	1	27992	~46 (!)	van
5	3	5	7240	124 (!)	van
6	2	2		550 (!)	van

* A mért idő függ a gép általános terhelésétől, többek között attól, hogy hány másik terminálon dolgoztak még.
** C64-esen kb. 20 perc a futási idő.
*** C64-esen kb. 3-4 perc.

A nagy gépet is megdolgoztató 6×6-os tábla megoldása (2,2) kiinduló koordinátákkal. (8. ábra)

tömb, mely összesen annyi elemet tartalmaz, amennyi az út sikeres bejárásához szükséges lépések száma. Ezt BASIC-ben pl. így deklaráljuk:

```
.
1420 dim b (n²)
.
.
.
```

A b tömbben minden eddig megtett (még sikert ígérő) lépéshez hozzárendelünk egy-egy számot, amelyek azt mutatják, hogy abban a helyzetben hány lehetőséget (irányt) próbáltunk ki. Mindaddig, amíg nem léptünk sehol vissza, a b tömb felhasználásával s-edik állásban a huszár következő lépésének koordinátáit pl. így irhatjuk fel:

```
.
.
.
2350 x=x1+c (b(s),1)
2355 rem c a lépéstáblázat
2420 y=y1+c (b(s),2)
2425 rem s az állás száma
2340 b(s)+1
2345 rem feljegyeztem a soron következő irányt.
```

A teljes program listája a szöveg után található. (Figyelem! Az egyes gépek BASIC-tájszólásai különbözhetnek egymástól. A közölt, kipróbált programot „értelmesen" kell adaptálni.)

A program összesen 132 sorból áll.

100—1818 Ez az „előkészület" rész: meghatározzuk a sakktábla méretét (n), a huszár kiinduló koordinátáit (x_0, y_0), beállítjuk a sakk szabályainak megfelelően a lépéstáblázatot (c), létrehozzuk azt a tömböt, amelyben minden egyes lépéshez tároljuk a próbálkozás sorszámát (b), létrehozzuk a sakktáblát a gépben (a) stb.

1960—3525 A 3450. sorban megnézzük, hogy nem értünk-e már a feladat végére. Ehhez összesen (n^2-1) lépést kell megtennünk. Ha még nem tartunk a végén, akkor megnöveljük a lépésszámlálót ($s=s+1$), majd a 2340. sorban kiválasztjuk a következő lépést, és a 2350., illetőleg a 2420. utasításban generáljuk az ehhez tartozó mező koordinátáit.

A 3310—3350 utasításokban egyrészt megnézzük, hogy nem lépünk-e le így a tábláról, illetőleg nem jártunk-e már előzőleg ezen a helyen ($a<>0$). Ha a lépés elfogadható, akkor feljegyezzük a táblára (3427.), és visszatérünk a továbbléptetés elejére (3525).

9000—9500 Ez a visszaléptető rutin. Természetesen visszaléptetni csak legalább a második lépéstől értelmes (9050). Ha visszalépünk, akkor egyrészt törölnünk kell a táblán a korábbi bejegyzést (9100), másrészt vissza kell állítanunk a próbálkozásszámlálót nullára (9080). Ezután a lépésszámlálót is csökkentjük eggyel (9200), és segítségével megkapjuk az első lépés koordinátáit (9310, 9320). Itt c értékét nem hozzáadjuk a jelenlegi koordinátákhoz — mint előreléptetésnél tettük —, hanem levonjuk belőle, majd visszatérünk az új lépés generálásához: a visszaléptetett pozícióból *új* úton fogunk elindulni.

11000— Az eredmény megjelenítése

A k változóban az összes megtett lépést és próbálkozást számoljuk, és értékét a végén megjelenítjük. Érdemes megfigyelni, hogy ugyanakkora táblán különböző pozíciókból kiindulva mennyire különbözik a megoldáshoz vagy a megoldhatatlanság kimutatásához szükséges lépések száma. Tapasztalhatjuk azt is, hogy a gép (illetőleg a program) nem ismeri fel a szimmetriát: pl. a tábla különböző sarkaiból indulva különböző megoldásokhoz jut el eltérő lépésszámokkal.

„Szorgalmi" gyakorlatként kínálkozik az algoritmus továbbfejlesztése úgy, hogy egy-egy kezdő pozíciótól kiindulva ne csak egy megoldást keressünk meg, hanem az esetleges többit is; a fejtörést az olvasóra bízzuk.

Bár a program viszonylag rövid, a futási idő táblamérettől és az indulás helyétől függően otthoni gépeken (pl. C64-esen) igen hosszú (több óra vagy akár több nap is) lehet. Már a 8×8-as méretű táblával való kísérletezést sem ajánljuk otthoni gépeken. (6×6-os táblánál is elviselhetetlenül hosszú futási időket kaphatunk.) Az alábbiakban megadjuk néhány esetre a megoldáshoz vezető lépések számát, a CPU-időt (melyet egy nagy teljesítményű gépen mértünk) és azt, hogy van-e az adott esetben megoldás.

i	z	
	1	2
1	1	2
2	2	1
3	2	−1
4	1	−2
5	−1	−2
6	−2	−1
7	−2	1
8	−1	2

Az érdeklődőknek agytornaként javasoljuk, gondolkozzanak el, mi az oka, hogy

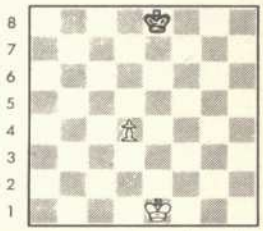
— ha nincs megoldás, akkor szimmetrikus kiindulási pontoknál azonos számú lépés kell annak megállapításához, hogy nincs megoldás, míg

— ha van megoldás, akkor szimmetrikus kiindulások esetén (pl. ilyen az 5×5-ös táblán az [1,5], ill. az [1,1] koordinátapár) a megoldáshoz vezető lépésszámok eltérőek. Jó töprengést kívánunk!

— KE —

BITEK ÉS FIGURÁK

Végjátékok I.



Capablanca a játszma legfontosabb részének tartotta a végjátékot.

Már a kezdő sakkozó is hamar meggyőződhet arról, hogy a játszmát milyen nehéz a végjátékban megnyerni. A középjáték bonyodalmai ugyanis általában köznapi értelemben is bonyodalmasak, tehát nem nagyon eredményeznek kész helyzetet — nem teszik lehetővé az ellenfél „túljátszását" —, s ezért egyenlő vagy közel egyenlő ellenfelek között a játszma sorsa leggyakrabban a végjátékban dől el. A megnyitás és a középjáték küzdelmei során az egyik fél legtöbbször csak egészen kis előnyt szerez. Egy gyalog- vagy egészen kis pozíciós előnnyel evez a fölülkerekedő a végjáték bizonytalan vizeire. Itt kellene a parányi előnyt a győzelemig fokozni. A végjáték-szituációk pontos ismeretével hátrányos helyzetből is meg lehet fordítani a parti sorsát, pontatlan végjátékvezetéssel pedig az előnyösebb állást is elronthatjuk.

Kisebb megnyitási hibát vagy a középjátékban elkövetett pontatlanságot később talán sikerül szerencsével és ötletes játékkal kijavítanunk, azonban a végjátékban elkövetett hiba majdnem mindig végzetes. Nem véletlen, hogy a világbajnokok kivétel nélkül kiváló ismerői voltak a játszma e szakaszának. A számítógépek viszont sajnos kevésbé ismerik fortélyait: a programozók számára a legfőbb gondot okozzák. Ennek magyarázata egyszerű: túl messzire kell előre számítani ahhoz, hogy az elágazások exponenciálisan növekvő száma miatt — a lecsökkent anyag mellett is — a gép minden változatot képes legyen értékelni.

Melyek a végjáték legjellegzetesebb tulajdonságai?

1. A leglényegesebb a király szerepének megnövekedett jelentősége. A végjátékban királyunkat csak ritkán fenyegeti közvetlen mattveszély, s így aktív szerepet tölthet be; szükség is van rá, hogy támadó váljon belőle. Az a fél, amelyiknek aktív lehet a királya, míg ellenfelének királya korlátozva van a tevékenységében, úgyszólván tiszta előnnyel játszik.

2. A végjáték másik főszereplője a gyalog. A leggyakrabban a gyalog átváltozása, illetve az átváltozás megakadályozása az ellenfél részéről a küzdelem tulajdonképpeni célja. A gyalogállás minősége, szerkezete döntő fontosságúvá válik, mint ahogyan ezt sorozatunkban előzőleg tárgyaltuk.

3. A végjátékban érvényesül legjobban a törvényszerűség. A végjáték elvei, szabályai időtállóak. (Néhány elemmel bíró rendszer állapotai „kiszámíthatóbbak". — A szerk.)

A hadállás megítélésének képessége a jó sakkozó leglényegesebb tulajdonsága. A végjátékismeret lehetővé teszi, hogy a középjáték bonyodalmai közepette is távlatokban gondolkozzunk. Sajnos a gépeknél ez teljesen másként van. Míg a sakkozó ember az állás változásait folyamatokban vizsgálja, és minden egyes állást dinamikusan értékel, addig egy számítógép a különböző feltételek teljesülésekor (pl. ha a lépésszám elér egy korlátot) egyszerűen egy másik értékelőfüggvénnyel folytatja tovább a számítást. Emiatt a játszma egyes részei teljesen elkülönülnek egymástól. Ezért a középjáték és a végjáték közötti átmeneti játszmaszakaszban korunk sakkprogramja még nem képes figyelembe venni a végjáték alapelveit, és a végső lebonyolítás sokszor a gép hátrányára sikerül. A programozók napjainkban ezért egyre jobban koncentrálnak a középjáték és a végjáték közötti szakasz értékelőfüggvényének javítására.

Gyalogvégjáték

Különösen nagy jelentőséget kell tulajdonítani a GYALOGVÉGJÁTÉK-nak, amelyben a királyok mellett csak gyalog(ok) szerepel(nek).

Megfigyelhetjük, hogy gyalog nélküli végjáték csak a legritkább esetben fordul elő. Miért van ez? Minden végjátéktípus alkalmas pillanatban átalakulhat ugyanis gyalogjátékká. Az erősebbik fél számára gyakran épp az ilyen lebonyolítás igéri a legegyszerűbb utat a győzelemhez, vagy fordítva: a gyengébb fél sokszor gyalogvégjátékban keres menekülést szorongatott helyzetéből. Tudni kell tehát, hogy egy kialakuló gyalogvégjáték melyik fél számára előnyös! A jó sakkozó számára ez ritkán gond.

A gyalogvégjátékok a végjátékismeret alapját alkotják, tehát az elveket és a törvényszerűségeket viszonylag egyszerűen be lehet programozni.

Ha a gyalog a saját királyától távol áll, átalakulása csak úgy lehetséges, ha az ellenfél egyedülálló királya ebben nem tudja megakadályozni. Ezt egyszerű számítással meg tudjuk oldani, de a gyorsabb számítás érdekében alkalmazhatjuk az ún. négyzetszabályt. Ha a gyalog kiindulási pontjától az átváltozási mezőig megszámoljuk a mezőket, majd ebből fejben egy négyzetet alakítunk, „ránézéssel" megállapíthatjuk, hogy az ellenséges király utolérheti-e a gyalogot vagy sem: ha a védekező fél királya ugyanis a négyzeten belül foglal helyet, vagy mint soron következő, a négyzetbe léphet, akkor utolléri a gyalogot és lenyeri. Fontos, hogy az eredeti helyén álló gyalognál a négyzetet úgy kell elképzelni, mintha a gyalog egy mezővel előbb állna, mert a gyalog a következő lépésben két mezővel is előreléphet.

A négyzetszabályt matematikailag a következő formulával fejezhetjük ki:

T = max (gyal. sor — ell. kir. sor, gyal. oszl. — ell. kir. oszl.)

A képlet rövidítései:

gyal.: gyalog
ell.: ellenséges
kir.: király
oszl.: oszlop

Mivel mindegy, hogy két pont között a királlyal egyenesen vagy átlósan tesszük meg az utat, a legrövidebb úton pontosan annyi átlós lépéssel lehet a királlyal elérni a gyalogot, amennyi az oszloptávolságuk és a sortávolságuk maximuma.

Csak ritkán fordul elő, hogy a gyalog királya támogatás nélkül is célt ér. Az ilyen ritka állásokban döntő jelentősége van a mértani oppozíciónak, a szembenállásnak, amelyről már szintén szó esett.

Beszélhetünk távoli és közeli oppozícióról. A gyalogbevitelnél leggyakrabban a közeli oppozíciót alkalmazzuk, de a távoliról sem szabad megfeledkeznünk. A következő képlet mind a távoli, mind a közeli oppozícióra vonatkozik: pozi-

tívumként értékeli, ha a királyok közelítenek egymáshoz, és megpróbálják egymást leszorítani.
(16 — [Δ oszloptávolság + Δ sortávolság])/2

Ha ennek a kifejezésnek az értéke páros, akkor fennáll az oppozició. Vagyis: ha az oszloptávolság és a sortávolság különbségének összegét 16-ból kivonjuk, az érték akkor lesz nagy, ha a két király közötti oszloptávolság és sortávolság kicsi: tehát, ha a két király megpróbálja leszorítani egymást, és ezzel közeli oppozíciót ér el. Erre a gyalogbevitelnél és mattadásnál is a legtöbb esetben szükség van.

A képlet használata előtt győződjünk meg arról, hogy a programban helyesen használjuk-e: ugyanis egyáltalán nem mindegy, hogy a lépésre következő fél szempontjából vesszük-e figyelembe, vagy a védekezésnél. Mindkét eset figyelembevétele a legjobb megoldás . . .

Hogyan biztosítsuk gyalogunk bevitelét? Milyen tervet alkossunk? Végcél, hogy királyunkkal elérjük a gyalog melletti egyik oszlop 7. (illetve sötéttel 2.) mezejét. Például világos d3 gyalog bevitelénél világos királyunkkal a c7 vagy az e7 mezőt kell elérni.

Nem tudja-e azonban sötét király a tervet megakadályozni? Tudnunk kell, hogy minden szabad gyalog bevezetésének vannak bizonyos kulcspontjai. Ezek egyikének elfoglalása biztosítja a gyalog előnyomulását. A saját térfélen lévő gyalog esetében szám szerint három ilyen kulcspont van: egy a gyalog előtt két mezővel és egy-egy attól jobbra, illetve balra. Esetünkben (lásd az *ábrát)* ezek: d5, c5, e5.

Mit jelent ez közelebbről? Nem keveset: ha világos királyunkkal sikerül e pontok bármelyikét elfoglalni, akkor függetlenül attól, hogy hol áll a sötét király és melyik fél van lépésen, a gyalog bevitelét nem lehet megakadályozni. A helyes játék: 1. Kd2, Kf7; 2. Kc3, Ke6; 3. Kc4!, Kd6; 4. Kd4, Kc6; 5. Ke5!, Ke7; 6. Kd5 stb.

Ezt a sakkprogramban úgy valósíthatjuk meg, hogy bizonyos plusz pontot adunk arra az állásra, amelyben a gyaloggal rendelkező fél királya elfoglalta a kulcsponti mezők egyikét. Itt azonban jelentős probléma merülhet fel. Mi van akkor, ha a király már a kulcsponti mezőn van (az előbbi ábra szerint például d5-ön), és éppen ezzel a plusz ponttal akadályozzuk a gyalog bevitelét, mivel a kulcsponti mezőt nem akarja elhagyni? A lépéssor figyelembevételét a dinamikus értékelőfüggvénybe kell beépíteni, így a program könnyen figyelheti az egymás után következő lépéseket. Ha már olyan állást elemzünk, melyben a király elfoglalta a kulcspontját, akkor ennek a mezőnek vegyük le a súlyozását, és a gyalogbevitel egyéb szempontjait vegyük figyelembe.

Ha a gyalog túljutott a félvonalon — vagyis elérte az 5. sort —, akkor a kulcspontok száma megkétszereződik, mert nemcsak a gyalog előtti második, hanem közvetlenül a gyalog előtti soron is 3-3 mező számít kulcspontnak. Például ha d5-ön áll a gyalog, akkor kulcspontjai a c6, c7, d6, d7, e6, e7 mezők (lásd az ábrát).

Más típusú gyalogállások kulcspontjait szintén érdemes megvizsgálni, és minél többnek a szabályszerűségét sikerül az algoritmusba beépíteni, annál nyerőbb a helyzet.

KOVÁCS P. ATTILA

Megalakult a jugoszláv Atari klub.

Cím: MIKRORAČUNARSKI KLUB BEČEJ
"ATARI BIT"
21220 BEČEJ Pf.: 102
JUGOSZLÁVIA

ZX-SPECTRUM

CSIPP-CSEPP

Ez a játék nagyon jó reflexeket kíván, és fejleszti is azokat. A program RUN parancsra indul, majd a képernyő tetejéről esőcseppek potyognak. A játéktér négy számozott részre van osztva. Attól függően, hogy az esőcsepp melyik számú részben jelenik meg, le kell nyomni az adott számnak megfelelő billentyűt. Ha elég gyorsak vagyunk, tíz csepp elkapása után a játék nehezebb fokozatra kapcsol, ami azt jelenti, hogy a csepp „közelebbről" esik. Így a játék folyamán egyre nagyobb figyelemre és összpontosításra van szükség. A fokozatszám a képernyő bal oldalán, az el nem kapott esőcseppek száma a jobb oldalon jelenik meg.

Tíz hiba után „BETELIK A POHÁR".

PÉCSI RICHÁRD

```
  1 GO TO 5000
 10 LET x=0: LET y=10
 11 LET jo=0
 12 hiba=0
 13 nehez=0
 20 LET x=x+1
 30 IF x=19 THEN GO TO 1000
 40 IF INKEY$="4" THEN PRINT AT
19,7;"_______": GO TO 7000
 45 IF INKEY$="2" THEN PRINT AT
19,0;"_______":GO TO 7000
 46 IF INKEY$="6" THEN PRINT AT
19,14;"_______": GO TO 7000
 47 IF INKEY$="8" THEN PRINT AT
19,21;"_______": GO TO 7000
 50 PRINT ATx,y; BRIGHT 1;"$";
AT x-1,y;" "
 60 GO TO 20
1000 PRINT AT x-1,y;" ": LET x=n
ehez
1010 LET y=INT (RND*25) +1
1015 IF jo=10 THEN LET nehez=neh
ez-1: LET jo=0: LET hiba=0: PRIN
T AT 12,0; nehez: FOR u=0 TO 15:
PRINT AT u,29;" ";AT u+1,29;" ":
 NEXT u
1020 PRINT AT 19,0;"

1030 GO TO 9000
5000 LET nehez=0:BORDER 0:PAPE
R 0:INK 7: CLS : FOR i=0 TO 7:
READ q: POKE USR "a"+i,q: NEXT i
5010 DATA BIN 00011000,BIN 00011
000,BIN 00111100,BIN 01111110,BI
N 01111110,BIN 01111110,BIN 0011
1100,BIN 00011000
5500 FOR r=0 TO 28 STEP 7: PRINT
 AT 20,r;" ":NEXT r
5510 PRINT AT 20,3;2;AT 20,10;4;
AT 20,17;6;AT 20,24;8
5560 PRINT AT 21,8;BRIGHT 1;"Pe
csi Rihard";AT 20,9;",";AT 20,1
8;","
5570 PRINT #0;"        CSIPP-CS
EPP"
5580 PRINT AT 12,7;"0-ra indul a
 jatek": IF INKEY$<>"0" THEN GO
TO 5580
5590 PRINT AT 12,7;"
                "
5595 PRINT AT 10,9;"
    ";AT 9,15;"          "
5596 FOR u=0 TO 18: PRINT AT u ,2
9;" ";AT u ,30;" ":NEXT u
5597 PRINT AT 12,0;"  "
6000 GO TO 10
7000 PRINT AT x,y; BRIGHT 1;"$";
AT x-1,y;" "
7005 IF SCREEN$ (x+1,y)=" " THEN
 BEEP .1,40: LET jo=jo+1: LET hi
ba=hiba-1: PRINT AT x,y;" ":GO
TO 1000
7010 LET x=x+1
7020  IF x=19 THEN PRINT AT x-1,y
;" ": GO TO 1000
7030 GO TO 7000
9000 LET hiba=hiba+1
9005 PRINT AT 12,29;hiba
9010 PRINT AT 10-hiba,29;" "
9020 IF hiba=10 THEN PRINT AT 10
,9; FLASH 1; BRIGHT 1;"BETELT A
POHAR"; FLASH 0;AT 9,21;",";GO
TO 5580
9030 GO TO 20
9999 REM CSIPP-CSEPP Pecsi Richa
rd
```

COMMODORE 64

Apróságok

Ezt a három programot kezdőknek ajánljuk. Az első egy ágyú hangját szólaltatja meg, a második erre ad visszhangot. A SID chip regisztereinek ismeretében könnyen változtathatunk a DATA sorokban megadott értékeken. Mindkét program végtelenített.

Tenisz lehetne a harmadik program neve. A tanulás kezdetén mindenki megpróbál hasonló játékot alkotni. Mi azzal a kikötéssel fogtunk hozzá, hogy terjedelme nem haladhatja meg a tíz sort, és ennek ellenére működő játék legyen.

A H változó a hibajelző, értékének növelésével a játék hosszabbítható. Hibajelzésként a keret színváltozása érzékelhető. A labda minden visszapattanása — az oldalakról is — egy-egy pontot ér. Az ütő hosszát a 3-as sor PRINT utasításában lévő szóközök száma határozza meg. Az ütő a vessző és a pont billentyűkkel mozgatható.

FÖLDI ENDRE

Várjuk mások hasonló próbálkozásait, azaz kérjük a kedves olvasót, hogy írjon minél rövidebb, működő játékprogramot! (A szerk.)

```
10 REM TAVOLBAN AGYUZNAK
20 SID=54272
30 FORX=0TO24:READA:POKESID+X,A:NEXTX
40 DATA  0,  6,  0,  0,128,  1,252,  0
50 DATA  0,  0,  0,  0,  0,  0,  0,  0
60 DATA  0,  0,  0,  0,  0,  0,  6,241
70 DATA 47
80 POKESID+4,129:POKESID+4,128
90 FORX=0TO3000:NEXTX:GOTO80
100 POKESID+22,PEEK(SID+27)
110 NEXTX:GOTO80
```

1. program

```
10 REM VISSZHANG
20 SID=54272
30 FORX=0TO24:READA:POKESID+X,A:NEXTX
40 DATA  0, 15,  0,  5, 16,  1,252,  0
50 DATA  0,  0,  0,  0,  0,  0, 60,  0
60 DATA  0,150, 17,  0,  0,  0,  0,  9
70 DATA 31
80 POKESID+4,129:POKESID+4,128
90 FORX=0TO400
```

2. program

3. program

```
0 X=1:Y=1:XL=1:YL=1:POKE650,128:
  PRINT"■":
1 GET#6:IF6$="","THENIFXU<>36THENXU=XU-2
2 IF6$="."THENIFXU<>36THENXU=XU+2
3 POKE211,XU:POKE214,23:SYS58732:
  PRINT"■■  ■  ■  ";
4 IFXL=0ORXL=39THENX=-X
5 XL=XL+X:IFYL=1THENIFNOTYL=21THEN8
6 P=P+1:Y=-Y:
  IFYL=0ORXL>=XUANDXL<XU+5THEN8
7 H=H+1:POKE53280,H:IFH=6THEN
  PRINT"■AME OVER":PRINT"SCORE:"P:END
8 YL=YL+Y:POKE211,XL:POKE214,YL:
  SYS58732
9 PRINT"■■  ■■■■  ■■■■  ■■■■";:GOTO1
```

Szellem ellen szellemesen

A programmal a *nem automatikus* indítású, *lemezen* lévő, a kazettapuffert *nem használó* játékprogramok szellemei után kutathatjuk végig a memóriát, és közülük tetszőleges sokat törölhetünk. Így könnyebb a játék. A program három fő részből áll.

Az első rész (10—35-ös sorok) egy képernyőt állít elő, amelynek bal oldalán két keret, ablak van. Ide kerül majd a szellem fekete-fehér és színes megjelenítésben. Mellette látható a három aktuális parancs: F1 lapoz, F2 töröl és lapoz, Q pedig a programból való kilépést biztosítja. A képernyő alsó részén a használatot megkönnyítő parancsok vannak. A 10-es sor inverz záró idézőjele csupán a nyomtatáshoz használt EPSON illesztőjének hibája. Oda normál idézőjel kell!

A második rész (100—115-ös sorok) az SPR nevű programfájlba, lemezre menti a gépi kódú programot és a közvetlenül utána kezdődő, az első részben létrehozott képernyőt. Vizsgálja és kezeli akár a mentés, akár a betöltés közben előforduló esetleges hibákat.

A harmadik rész (200—225-ös sorok) a gépi kódú program DATA soraiból áll.

A közölt lista begépelése és sikeres futtatása után tehát lemezen rendelkezésünkre áll egy SPR nevű program. Ezután a szellemek törlésének menete:

1. Töltsük be a játékprogramot, de ne indítsuk el.

2. LOAD"SPR",8,1 paranccsal töltsük be az SPR programot, lehetőleg úgy, hogy a parancsot a képernyő alulról számított hatodik sorában adjuk ki; így a betöltés végén megjelenő READY felirat nem rontja el a képernyőt. Ekkor (indítása nélkül!) a képernyő a 10—35-ös sorokban megfogalmazott első programrész által meghatározott tartalmú lesz, és a kurzor villog (lásd fentebb).

```
10 POKE53280,11:POKE53281,11:POKE646,1
11 PRINT"■CHR$(8);:POKE53272,21
12 PRINTTAB(21)"SZELLEM TORLO"
13 PRINTTAB(5)"■SZELLEM";:REM CRSR LE
14 PRINTTAB(20)"BY EDESOFT 1986"
15 PRINT" ┌──────┬──────┐":REM SHIFT+A
16 PRINT" │      │      │":REM SHIFT+C
17 PRINT" │      │      │";:REM SHIFT+S
18 PRINT"   F1 -->LAPOZ":REM SHIFT+B
19 PRINT" │      │      │";:REM SHIFT+Z
20 PRINT"   F2 -->TOROL+LAPOZ"
21 PRINT" │      │      │":REM SHIFT+X
22 PRINT" │      │      │";
23 PRINT"   Q  -->KILEPES"
24 PRINT" └──────┴──────┘"
25 PRINT"   NORM   MULT"
26 T$="       ":REM 7 SPACE ES CRSR LE
27 PRINTT$"■──────────────────────────"
28 PRINTT$"SYS (828)      :REM INDITAS"
29 PRINTT$"POKE 253,0     :REM CIM LO"
30 PRINTT$"POKE 254,8     :REM     HI"
31 PRINTTAB(21)":REM SZINEK"
32 PRINTT$"POKE 53287,1   :REM NORM #0"
33 PRINTT$"POKE 53288,1   :REM      #1"
34 PRINTT$"POKE 53285,12  :REM MULT #0"
35 PRINTT$"POKE 53286,15  :REM      #1"
36 :
100 REM *** KIMENTES ***
101 OPEN15,8,15:OPEN8,8,8,"@:SPR,P,W"
102 PRINT#8,CHR$(60)+CHR$(3);
103 FORC=828TO1023
104 READA:PRINT#8,CHR$(A);:GOSUB110
105 S=S+A:NEXT:IFS<>24953THEN114
106 FORC=1024TO2023
107 PRINT#8,CHR$(PEEK(C));
```

COMMODORE 64

3. Vigyük a kurzort a SYS (828) parancsra, és nyomjunk RETURN-t. Ezzel elindítottuk az SPR programot, és a két ablakban megjelenik a 2048-as címtől kezdődő 63 bájt tartalma, mintha az egy szellem lenne! F1-gyel lapozhatunk a következő 64 bájtra. Ha ráismerünk egy nem kívánatos szellem egyik fázisrajzára, akkor ezt F2-vel törölhetjük, és egyben lapozunk is. Vigyázat, egy szellemnek általában több fázisrajza van. Ahhoz, hogy a szellem játék közben végképp eltűnjön, valamennyi fázisát törölni kell!

4. A képernyő úgy van szerkesztve, hogy a parancsok végrehajtása után megjelenő READY felirat nem töröl bele a megjelenített parancsokba. Ha a sor elején már szerepel egy előzőleg megjelent READY felirat, természetesen azt a parancs kiadása előtt törölnünk kell, különben hibajelzést kapunk, és ez elrontja a képernyőt. A SYS utáni két POKE utasítás értékeinek (nem címeinek!) megváltoztatásával a keresés kezdetét állíthatjuk be. Ha olyan kezdőértéket adunk, amely nem osztható 64-gyel, akkor az ablakban egyszerre két szellem bal és jobb oldali részlete lesz látható. Az utolsó négy POKE paranccsal a megjelenített szellem színeit változtathatjuk meg, állíthatjuk be.

5. Q-val kilépve az SPR programból, RUN-nal indítható a játék, amelyből már hiányoznak a törölt szellemek. Q-t kicsit hosszabban tartsuk nyomva, hogy a kilépés biztosan sikerüljön.

Figyelmeztetés: csak akkor töröljünk egy szellemet, ha egyértelműen ráismertünk. A program ugyanis folyamatosan nézi végig a memóriát, sőt ha elérte a végét, az elejéről, a nullás laptól kezdi újra, így meggondolatlan törléssel beleérhetünk a játékprogramba, vagy fontos adatokat, vektorokat stb. tehetünk tönkre!

A programot eddig sikeresen alkalmaztuk a China Miner, a Son of Blagger és a Hunch Bach játékok egyes alakjainak törlésére.

ÉNEKES FERENC

```
108 GOSUB110:NEXT:GOSUB109:END
109 CLOSE8:PRINT#15,"I":CLOSE15:RETURN
110 INPUT#15,A1,A2$,A3,A4
111 IFA1=0THENRETURN
112 GOSUB109
113 PRINT"DISK ERROR:"A1;A2$;A3;A4:END
114 PRINT"HIBA VAN A DATA SOROKBAN!..."
115 GOSUB109:LIST 200-219
116 :
200 REM *** GEPI DATA ***
201 DATA234,234,234,169, 11,141,248,  7
202 DATA141,249,  7,169, 81,141,  1,208
203 DATA141,  3,208,169, 40,141,  0,208
204 DATA169, 95,141,  2,208,169, 11,141
205 DATA 32,208,141, 33,208,169,  1,141
206 DATA 39,208,141, 40,208,169, 12,141
207 DATA 37,208,169, 15,141, 38,208,169
208 DATA  3,141, 23,208,141, 29,208,141
209 DATA 21,208,169,  2,141, 28,208,234
210 DATA169,  0,133,252,169,  0,133,253
211 DATA169,  8,133,254,120,169,158,141
212 DATA 20,  3,169,  3,141, 21,  3, 88
213 DATA 96,  0,165,252,201, 16,208, 75
214 DATA169,  0,133,252,166,203,224, 62
215 DATA240, 51,224,  5,240,  8,224,  4
216 DATA240, 15, 24,144, 24,  0,160,  0
217 DATA152,145,253,200,192,64,208,249
218 DATA234, 24,165,253,105, 64,133,253
219 DATA144,  2,230,254,234,160,  0,177
220 DATA253,153,192,  2,200,192, 64,208
221 DATA246, 76, 49,234,234,120,169, 49
222 DATA141, 20,  3,169,234,141, 21,  3
223 DATA 88,  0,234,230,252,024,144,221
224 DATA  0,234,234,234, 69, 68, 69, 32
225 DATA 83, 79, 70, 84
```

A Sinclair-tulajdonosok változatlanul elégedetlenek: keveslik a cikkeket, programokat, ismertetőket. Legutóbb egy QL-tulajdonos hívott fel, hogy Spectrum-cikkeket még csak-csak közlünk, de QL-t nem, pedig a legutóbbi árleszállítás óta megnőtt a QL-esek tábora. Már szervezetük is van: a QLub. Kedves Sinclair-követők, lássuk az írásokat!

Dani Zsolt, Chip software

Heves, Kun Béla út 17. 3360

és Lakatos János, Heves, Bródy Sándor u. 21. 3360

Ez már nem az első levelem önökhöz, bár korábban csak „Az Olvasó írja" rovatnak írtam.

Az 1986/11-12. számban olvastam, hogy munkatársakat keresnek a lap szerkesztőségéhez. Én 2 évvel ezelőtt kezdtem foglalkozni a PRIMO-val, és mit mondjak, megszerettem. Gépi kódban egészen jól kijövünk egymással. Csak egy a bökkenő. Sajnos nincs se számítógépem, se pénzem rá.

Gondolom, tudnának nekem segíteni, hogy kaphassak egy PRIMO-t – ha lehet, akkor egy A64-es típusút –, cserébe egy barátommal szívesen küldenék minden számhoz BASIC és Z80 gépi kódú programokat. Sajnos a forrásfájlokat kinyomtatni nem tudjuk, még soha nem dolgozott egyikünk sem nyomtatóval, de szeretnénk. Talán lesz rá mód. Az ötletek már megvannak, de eddig gép hiányában megvalósítani nem tudtuk. Remélem, az Önök segítsége által valóra válhatnak álmaink a programokkal kapcsolatban. Az Önök lapjában szívesen vennénk részt egyszerűbb, ügyes és kevésbé ügyes programokkal, melyek a felhasználók várakozásait számítógépükkel kapcsolatban részben kielégítenék. Ha minden számban megjelenhetne 1-2 oldalon az általunk beküldött program, azt hiszem, Önök szerint is jó befektetés lenne egy PRIMO-ért. Hangsúlyozom, hogy ez *szerintem* van (vagy lesz) így.

Az ilyen ajánlatra szokták azt mondani, hogy „lássuk a medvét!" Küldjön írásokat, programokat, amelyeket – ha jók – közlünk, és a szokásos honoráriumot fizetjük. Gépet – sajnos – adni nem tudunk, még úgy sem, ha ledolgozzák.

Király László, Sömmerda-Fichter 15.

Pf.: 005-03. 5230. DDR

Szemmel láthatólag csökkentették a hosszabb programok leírását. Ezzel szemben megemelték a sakkal foglalkozó oldalak számát. Nem mindenki szeret sakkozni, de mindenki szeret játszani. Sajnos az egyes programokról szóló leírások nagyon rövidek. Mindenki tudja, hogy a profi programok hivatalos leírásai sem elég részletesek, ezért munka közben igen sok probléma merülhet fel. Szívesen olvasnék olyan cikket is, mely „nagyobb programok" hiányosságára mutat rá.

Levelét és tanácsait köszönöm.

Kovács Győző, aki e rovatot személyesen gondozza, itt tartott a levelek megválaszolásában, amikor lábtöréses balesetet szenvedett. Ezért az alábbi olvasói leveleket a szerkesztőség válaszolja meg.

Pjeczka Attila, Albertirsa,

Koltói Anna u. 26. 2730

Kedvenc számítógépemről, a ZX—Spectrumról szeretnék megtudni egyet s mást. Érdekelne, hol lehet kapni és menynyiért? A számítógépről szóló könyveket hol vásárolhatnám meg? És egy Spectrum botkormány rajzát is keresem.

Szeretném megtudni, hogy az UDG programozott karaktereket hogyan lehet felhasználni a rajzoláshoz? (Erre a Csupa játék ZX—Spectrumra c. könyv utalt. [Szerk.: Votisky Zsuzsa, Műszaki Könyvkiadó, 1985., 77 old., 28,— Ft]) (Pl. Challenger, Bombázó).

ZX—Spectrumot Budapesten a Tanács krt.-i Műszaki Bizo-

mányi Áruházban, a TEMPO Bajcsy Zs. út 53. sz. alatti üzletében kaphat. Ára: 15—18 000,— Ft között ingadozik. A botkormány rajzát megküldjük önnek.

Spectrum szakkönyveket a Liszt Ferenc téri Műszaki Könyváruházban, a SZÁMALK Bp. XI. Szakasits Árpád u. 68. sz. alatti épületében, valamint a Novotrade RT-nél vásárolhat (XIII., Fürst S. u. 24/26.). Felhívjuk figyelmét a következő könyvekre – melyekben kérdéseire is megtalálja a választ.

Bozetti: ZX—Spectrum — Tippek és trükkök, DATA BECKER — Novotrade 160 old., 199,— Ft. Dr. Ada-Winter Péter—Ada-Winter Dávid: A ZX—Spectrum — Hardverleírás — Gépi kódú programozás. Műszaki Könyvkiadó 360 old., 57,— Ft.

Püskös László, Debrecen

Batthyány u. 7. 4024

... legnagyobb problémám, hogy nem tudom eldönteni, milyen gépet vásároljak. Ehhez kérném az Önök segítségét.

Tudjuk, hogy nehéz a döntés; a kiválasztáshoz néhány szempontra felhívjuk a figyelmét.

Nézze meg — a hazai árakat figyelembe véve —, hogy mit bír el a pénztárcája. Az anyagi lehetőségei engedte választékot aszerint szűkítse le, milyen feladatok megoldására szeretné használni a gépet, és e feladatokhoz melyik konfiguráció illik, de tájékozódjon arról is, hogy az óhajtott géptípusnak milyen a hazai programválasztéka. A lakóhelye környezetében (a városban) esetleg működő klub tagjai gyakorlati segítséget is adhatnak mindenben.

Herczeg József, Pécs

Enyezd u. 18. 7632

A Mikroszámítógép Magazint a hazai számítástechnikai szaksajtó választéka közül a legszínvonalasabbnak tartom. Érdekes témák kerülnek sorra a rovatokban, a programok is hasznosak. Gyakran van szükségem rá, hogy cikkeket keressek vissza régebbi számokban – gondolom másnak is. Így tűnt fel egy apróság, amit tanácsolni szeretnék: feltétlenül illene egy ilyen nívójú periodikához a tárgyév végén (vagy később) elkészített összesítő tartalomjegyzék.

Javaslatát már félig megvalósítottuk (sőt bizonyos értelemben túl is léptük): az 1987/2. számtól kezdődően minden számunkban tartalomleírást teszünk közzé a legismertebb külföldi szaklapok aktuális közleményeiről. Rövidesen kész lesz lapunk régebbi számainak feldolgozása is.

Vékony Sándor, Debrecen

Izsó u. 23. 4028

... Videoton TV—C tulajdonos vagyok, örülök, hogy ez a gép tud „magyarul".

A TV—Computer nagyon tetszik, jól használható, bár van néhány hibája. Ezek közül három mechanikai jellegű. A rúgós lábazat nagyon gyenge. Egy gomb erősebb lenyomásától már összecsuklik. Az antennakimenet szintén gyenge konstrukció, a géphez adott túl vastag koaxiális kábel alátámasztás nélkül sokszor kiesik belőle. A beépített botkormány pedig egy hónap után már eltört, nem ugrik vissza alaphelyzetbe. Célszerűbb lenne a kurzor mozgatását a Commodore Plus/4 nyilaihoz hasonlóan megoldani botkormány helyett.

A TV—C BASIC-jéből nagyon hiányzik a kör-ellipszis-sokszög- (és ezeket elforgatva is) rajzoló utasítás. A TV—C ROM-jában van még 4 Kbájt üres hely. Remélem, valamikor a VIDEOTON kiad majd EPROM-okat, amelyeket a gépben lévő két üres foglalatba ültetve ez a hiányosság megszűnik! Kényelmetlen még, hogy csak egyszólamú hangja van a gépnek. Ezen sem ártana segíteni. Amikor egy teljesen új konstrukciójú gépet vettem, tudtam, hogy sokáig nem lesznek hozzá programok, bár titokban reménykedtem, hogy hátha ...!

Más. Hiányolom a reklámból az *árakat*. Telefonszámok mellett ezeket is közölni kellene, mert többet mondhat az áruról, mint egy kétoldalas reklámszöveg.

Tetszenek a részletes programbemutatók. Ezekből több kellene, hogy ha véletlenül választani lehet a programok között, ez ne csak találomra történjen: az ember legalább körülbelül tudja, hogy az adott programok közül melyik mit ér, mire képes.

Természetesen (?) hiányolom a TV—C programokat. Eddig összesen kettő darab (!) ilyen gépre írt program jelent meg a Magazinban!

A TV—C-vel kapcsolatos értékes észrevételeit megküldtük a VIDEOTON fejlesztőinek is. Panaszával a TV—C programellátottságáról, illetve vonatkozó bírálatával egyetértünk, és reméljük, hogy a közeljövőben egyre több írás születik az említett gépre készített szoftverekről. Programjait, ötleteit szívesen fogadjuk.

Lőrinczy Zsigmond, Budapest

Cimbalom u. 1. 1025

Nagy érdeklődéssel olvastam a PROLOG nyelvről szóló írást (87/1), amely azonban kevés a nyelv megismeréséhez. Nem világos az összetett objektumok szerepe, létrehozása, a beépített állítások használata (egyelőre elképzelni sem tudom, hogyan alkalmazhatók aritmetikai relációk a leírt szövegkezelő rendszerben). A „!" (ovt) eljárás hatását (és jelentőségét) egyáltalán nem értem.

Kérem a Szerzőt, további cikk(ek)ben írjon a nyelvről, összetett példaprogramokkal mutassa be a PROLOG előnyeit.

A PROLOG nyelv bemutatására indítunk egy sorozatot, melyben valószínűleg választ kap felvetett problémáira is. Lásd továbbá könyvrovatunkat is!

Nagy Aladár, Érd, VII.

Tálya u. 7. 2030.

Az árak ismeretére úgy gondolom, több szempontból is szükség van. Részben, hogy a felhasználók legyenek tisztában a lehetőségeikkel, a gyártók, importőrök pedig érezzék a felelősségüket a jelenlegi hazai árak dolgában. Az országos elektronikai fejlesztési programot az érvényes hardverárak nem szolgálják. Adó- és vám-, valamint közgazdasági szabályozók alkalmazásával az árakat le kell nyomni, ez hosszú távon a gyártók, importőrök és felhasználók jól felfogott érdeke. (Ezzel persze semmi újat nem mondtam.) A téma állandó szem előtt tartása – az árlisták rendszeres ismertetése — a mieztartás végett hasznos lehet. A TV-híradóban alkalmazott „sokkoló" módszer talán itt is alkalmas lesz! Mondjuk, hívjuk így: „ársokkoló".

Egy megjegyzés a közlendő árakhoz: átlagárak felsorolása volna a leghasznosabb — NSZK viszonyok között. Az árak talán úgy mondanak még többet, ha nem az érvényes valutaszorzóval számolunk, hanem például egy 8088 CPU-t hány órai keresetből lehet nálunk és máshol megvenni.

Tessék csak számolni! Így még keményebben, sürgetőbben dörömbölne a „pozitív" irányú árváltozás negatív ... irányban.

Remélem, az árak közlése folytatásos cikk lesz!

Igen. A levelével jó kaput dönget: régóta terveztük, sokáig halogattuk (ügyes-bajos ürügyekkel), de a következő számban már keresheti „Áruk és árak" új állandó rovatunkat. Ez az itthon kapható (és várhatóan beérkező) keresettebb termékek bel- és külföldi árait fogja ismertetni. Bízunk benne, hogy izgalmas titkot „árultunk" el — s nem csak önnek ...

Asztalos István, Pilisvörösvár

Bécsi út 1. 2085.

A Mikroszámítógép Magazin 85/2. számában olvastam, hogy 50 Ft jelképes összegért bármilyen gyári botkormányt átalakít Primóhoz használható botkormányhoz. Szeretném elkérni azt a programot is, amelyik a 85/2. szám 28. oldalán az 1. képen látható.

Ha a gépe hátlapján van A és B jelű csatlakozó, akkor a gép átalakítása nélkül lehet a botkormányt csatlakoztatni.

Használhatatlan!

1986 utolsó napjaiban bukkant fel a könyvesboltok polcain az a kisalakú, ám nagyon drága kiadvány, amelynek fedelén a BASIC ZSEBKÖNYV cím olvasható. Ára 99 forint, ami a mai könyvárak mellett nem látszik soknak, de ez a kötet a tizedét sem éri meg. Ugyanis teljesen használhatatlan.

A szerkesztők az előszóban azt írják, hogy emlékeztetőnek szánják a könyvet: olyan tömör összefoglalónak, melyben utána lehet nézni valamely BASIC nyelvi elem (utasítás, parancs, függvény stb.) helyes alkalmazási módjának. Minden lapon felsorolják, hogy az ott ismertetett nyelvi elem az általuk kiválasztott géptípusok közül melyeken érvényesíthető. A típusokat az előszóban sorolják fel, ahol például a SIMON's-ról egy szó sem esik, az egyes lapokon viszont természetesen szerepel. Könnyen kitalálható, hogy a C64-es gépeken használható SIMON's BASIC-ről van szó. A géptípusok között szerepel a RAAB '84 is. Valószínűleg az én tájékozatlanságom az oka, hogy erről a nagyon elterjedt számítógépről eddig még nem hallottam. A Videoton TVC-jének viszont a szerkesztők nem jósolnak nagy jövőt — különben miért hagyták volna ki?

A könyvben hemzsegnek a hibák. Íme néhány furcsaság az első részből:

A 37. oldalon az áll, hogy a DRAW utasítás a C16-on nem működik. A 69. oldalon rálelünk a LINE utasításra, mely vonalat húz, de téglalapot is rajzol. A C16-on ezt DRAW-nak kell írni. Csak az a probléma, hogy amit a C16-on DRAW néven ismerünk, az pontot rajzol vagy vonalat húz, és a paraméterezése sem egyezik meg a 69. oldalon találhatóval.

Nem felel meg a valóságnak, hogy a STOP utasítás csak a C16-on él. Nem igaz az sem, hogy a RUN parancs csak a 113. oldalon felsorolt géptípusokon létezik. Az is biztos, hogy — a 120. oldalon leírtakkal ellentétben — a SWAP utasítást nem lelem sem a C64, sem a C16 BASIC-jében, és az általam kevéssé ismert SIMON's BASIC leírásában sem találkoztam vele.

A 116. oldalon az SCNCLR utasításnál fent azt látom, hogy minden géptípuson aktuális, a lap közepén viszont már azt olvasom, hogy csak a C16 használja, megfelel a szabványos BASIC CLS utasításának. Nosza, nézzük meg a CLS-nél! A leírásban nincs ellentmondás, csak a felsorolt géptípusok száma csökkent nyolcra.

Sokáig idézhetném a hibákat, mert sok hasonlóra bukkantam még. Az ábécérendbe szedett kulcsszavak sorrendjének megkeverése sem a jó tájékozódást segíti. Nem tudom kitalálni, hogy a WRITE # utáni oldalon milyen utasítás lehet, csak azt, hogy melyik öt géptípuson próbálkozhatom vele.

A második részt alkotó referencialapok sem hibátlanok. Mutatóba egyetlen példát említek a Commodore lapokról: a C—64-nél hibás a DEF FN utasítás szintaktikája, az FNxx függvény hiányzik; a C—16-nál éppen fordítva, a DEF FN hiányzik és az FNxx függvény van hibás szintaktikával leírva.

A kiadvány a MÉDEA Kisszövetkezet gondozásában jelent meg. Szerkesztői Kiss Ádám és Kiss Balázs, lektorai dr. Bakó András és Tringer Éva voltak. Nem tudom, hogy a terjesztését ki engedélyezte, de biztos, hogy meggondolatlan cselekedet volt. Jó lenne gátat vetni az ilyenfajta fércmunkák elszaporodásának.

BARNA LÁSZLÓ

Meggyesházi János–Pintér Tibor: ZX Spectrum haladóknak Felhasználói segédlet (Budapest, Műszaki Könyvkiadó, 1986. 199 oldal. Ára: 78,– Ft)

A kisszámítógépek egyre nagyobb számban jelennek meg környezetünkben: a munkahelyeken, az iskolákban, otthonunkban. Közöttük is az egyik legsikeresebb konstrukció a ZX-Spectrum.

A könyv nem gépismertető, nem tankönyv, nem a divatos módszerek summázata, hanem magas szintű összefoglaló a gépet használók széles táborának. A szerzők a géppel való minél kényelmesebb munkához kívánnak sokrétű, egyébként nehezen hozzáférhető információt adni. Kitérnek a gép néhány szoftver szempontból is lényeges hardver jellemzőjére, a BASIC programozás rejtett tudnivalóira, bővítési lehetőségeire, az assembly programozás mesterfogásaira. Hasznos, sokoldalú rutingyűjteményt közölnek, és szólnak két fontos hardver kiegészítőről is: az Interface 1 illesztőegységről és a Microdrive-ról, a Sinclair-cég speciális háttértárolójáról.

A Logo programozási nyelv Írták: Turcsányiné Szabó Márta–Senftleben, Dietrich. (Budapest, Műszaki Könyvkiadó, 1986. 317 oldal. Ára: 83,– Ft)

A könyv a külföldön már nagy népszerűségnek örvendő és hazánkban is rohamosan terjedő Logo programozási nyelv alapfogalmait, eszközkészletét, felhasználását ismerteti igen sok példával, dialektikus stílusban.

Sok személyi számítógéptípuson van Logo fordító: a könyv programpéldái a ZX Spectrum Logo változatában készültek, de a függelék több Logo változat összehasonlító táblázatát is tartalmazza.

Főbb fejezetek: Teknőc-grafika. — Üdvözöljük a Logóban (utasítások, nyomtató programok, műveletek definiálása stb.). — Logo haladóknak. — Példák nagyobb programokra. A függelékben az LCSI Apple Logo alapfunkcióit, a Commodore 64 Terrapin Logo leírását és a Sinclair ZX Spectrumra kidolgozott Snail Logót ismertetik a szerzők.

Geszti Péter–Kápolnai András: Az IBM PC BASIC fejlesztőrendszere (Budapest, 1986. LSI ATSZ, 414 oldal. Ára: 236,– Ft)

A könyvet szerzőik azoknak a programozóknak és rendszerfejlesztőknek ajánlják, akiknek programozási alapismereteik ugyan már vannak, de BASIC-ben még nem programoztak.

A könyv az IBM PC-ken és a velük kompatibilis számítógépeken elterjedt interpreter és fordító változatokat mutatja be, majd tankönyvszerűen írja le a BASIC 2.0 és a 3.0 interpretert, illetve a BASIC 1.0 fordítót. Ezután ismerteti a gép, az operációs rendszer, az interpreter és a fordító kezelését. Alfabetikus sorrendben közli az utasításokat, a függvényeket és a standard változókat.

A függelékben az olvasó táblázatokat talál a függvényekről, a képernyőkódokról, a billentyűzet kódjairól, és itt keresheti az interpreter hibakódjainak összefoglalását és a fordító hibajelzéseit is.

Kármán-nyomtató

Még 1983 tavaszán jelentették be Kármán Péter, a Kandó Kálmán Műszaki Főiskola tanára új elven működő mátrixnyomtatójának szabadalmát. A találmányból az elmúlt év végére lett termék: megjelent a solymári Rozmaring Mgtsz által előállított RPR 210—01 típusú nyomtató. Az új nyomtatási elv lényege, hogy a nyomófej nem tűkkel, hanem speciális mátrixlemezekkel működik. A dörzshajtású papírtovábbítás és az újszerű papírvezetés együtt sodrás- és gyűrődésmentességet, megbízható üzemet garantál. Vezérlése a beépített mikroprocesszor programozása, valamint az interfészeken keresztül kapott adatok és vezérlőkódok értelmezése szerint történik. A nyomtató 512, illetve 1024 pontos grafikus üzemmódjában mind a 8 függőleges mátrixpont elérhető. A szövegrészeket azonos pozícióban való rányomtatással lehet kiemelni. Festékszalagként 13 mm széles nejlon írógépszalag alkalmazható. A papírszélesség 206—222 mm között változhat; tekercspapír, leporelló és A/4-es méretű lapok egyaránt megfelelnek adathordozónak.

A nyomtató a karaktereket 9 × 8-as pontmátrixból képezi; 80 karakter/másodperc sebességgel nyomtat soronként maximum 80 leütést. A karakterszélesség kétszeresére is növelhető. Ez az eszköz műszaki paraméterei és illesztőegysége alapján elsősorban iskolai és házi célú mikroszámítógépekhez (Primo, ZX—Spectrum, Commodore), valamint professzionális személyi számítógépekhez ajánlott. A 8 kp súlyú nyomtató ára 35 ezer forint.

Mű vész-e a művészetre a mű mű?

Az Új Impulzus írta ki az első hazai számítógépes művészeti pályázatot azzal a céllal, hogy felmérje korunk e sokat ígérő alkotási szférájában a hazai esélyeket. A kezdeményezés minden előzetes várakozást felülmúlt: 50 pályázó 300 művet küldött be a felhívásra. A jelentkezők zöme technikakedvelő képzőművész, illetve művészetkedvelő műszaki. Volt, aki matematikai képlet szerint hozta létre alkotását, mások ecset-vászon kombinációként kísérleteztek a számítógéppel. A pályázókat — ha ezt kérték — az MTA SZTAKI térítésmentesen géphasználathoz segítette. A legsikeresebb alkotásokat végül méltó helyen, a Szépművészeti Múzeumban állították ki.

(Reméljük, IGEN!!!)

Élet, erő, egészség és a vonalkód

Úgy látszik, itthon és rendszeresen elsőként a tejipar fogja alkalmazni termékein a nyugati árukról már jól ismert vonalkódot. Ennek előkészítése az egységes termékazonosító rendszer jelenleg folyó kialakítása. A következőkben olyan elektronikus pénztárgépeket szereznek be, amelyekhez csatlakoztatható a vonalkód olvasására szolgáló fényceruza is. (Csak a fényceruza ki ne bökje majd a még addig legalább nem lyukas tejeszacskókat . . .)

A Veszprém megyei Tejipari Vállalat által készített Tihany camembert sajt dobozán már szerepel is a vonalkód, bár ezt még inkább csak az exportnak köszönhetjük.

KGST-színekben: színes nyomtató

A Plovdivi Perifériagyártó Kombinát új, nagy teljesítményű grafikus mikronyomtatót mutatott be. Az E—1200 nevű új mátrixnyomtató háromféle pontmátrix-mérettel működik, ennek megfelelően a sebessége is változik. 7 × 9 pontot 200 jel/s sebességgel ír, 14 × 9 pontnál felére csökken az írás sebessége, míg a 19 × 14 pontos nyomtatáshoz 40 jel/s érték tartozik. Színes nyomtatáshoz négyszínű szalagot kell belehelyezni, és ezáltal az összetett szín kialakítható. Grafikus üzemmódban az alternatívák: 72 × 72 pontos írás esetén 317 mm/s, 144 × 144 pontos írás esetén pedig 158 mm/s az elérhető teljesítmény. Párhuzamos illesztőjén kívül soros illesztője is van; ez maximum 9600 baudos átviteli sebességre képes.

Kímélő növényvédelem

Az NDK-ban, a Mezőgazdasági Tudományok Akadémiája Növénykutató Intézetében, Kleinmachnowban elkészítették a legveszélyesebb növénybetegségek matematikai modelljeit. Ezek elsősorban az időjárással való összefüggésben a lehetséges kórokozó-szaporodást veszik figyelembe.

Az NDK meteorológiai szolgálata naponta jelenti a 19 belföldi meteorológiai állomás időjárási adatait az intézet eberswaldei számítóközpontjába. A feldolgozás eredményét — a növényi kórokozók aktuális elterjedtségét és a következő két napban várható kórfejlődést — közlik a növényvédő állomásokkal. Ha a fertőzés mértéke meghalad egy határértéket, vagy a kórfejlődésben veszélyes trend ismerhető fel, a növényvédelem munkatársai megfelelő ellenintézkedésekről gondoskodnak.

A helyzet alapos ismerete lehetővé teszi, hogy a vegyi anyagokat pontosan, gazdaságosan és a környezetet kímélően adagolják. A SIMPYT modell alapján, amely a burgonyaszár rothadásának ellenőrzésére szolgál, például lehetővé vált a megfelelő gombaölő anyagok optimális adagolása, és ezáltal felhasználásuk más növényvédelmi feladatokra is. A gabonaliszthármat, illetve a szártörés elleni SIMERY és SIMCERC modell széles körű gyakorlati kipróbálása jelenleg folyik.

Szilícium-völgy Indiában?

India technológiai fejlődésének egyik új központjává vált a Kamataka állambeli Bangalore, ahol az utóbbi időben terebélyesedni kezdtek a csúcstechnológiákat alkalmazó iparágak. A ma 3,5 milliós várost India leggyorsabban növekvő települései között tartják számon, lélekszáma az évezred végén a prognózisok szerint 6 millió lesz.

Az első számítógép-összeszerelő üzemet még 1977-ben alapította egy Kaliforniából hazatelepült, a Stanford egyetemen végzett mérnök. Azóta több száz, jól képzett indiai jött vissza az USA-ból Bangalore-ba. Sikerült olyan „klímát" teremteniük a városban és környékén, hogy arra már több tucat neves külföldi cég is felfigyelt, és üzemet szándékoznak telepíteni oda. Az elsők az amerikai Hewlett—Packard és a Texas Instruments lesznek, de az indiai hatóságok előtt vannak már a holland Philips, az amerikai Data General és a svéd L. M. Ericsson engedélykérelmei is — videomagnók, számítógépek, illetve telefonberendezések szereléséről van szó.

Bangalore a hazai cégek kutatás-fejlesztési munkájában is egyre fontosabb szerepet tölt be. Az Indian Telephone Industries például üvegkábel- és lézergyártással kísérletezik itt. Egy szintén Amerikából hazatért számítógépes szakértő nemrégiben olyan iskolát nyitott a városban, ahol alapfokú számítástechnikai képzést kapnak a gyerekek.

64 bites 1990-re?

A francia Thomson cég irányításával fejlesztés alatt áll egy 64 bites mikroprocesszor. Csökkentett utasításkészletű (RISC) architektúrájún már 1981 óta dogoznak.

A hét integrált áramkört tartalmazó teljes rendszer teljesítőképessége megfelel 4 db VAX 11/780-nak, és várható teljesítménye 4—30 milliárd művelet/s lesz. A szoftvermagjában a Unix operációs rendszer is megtalálható majd, s az Ada fordítót már készítik hozzá.

A program voltaképpen hozzájárulás az Eurêka rendszerhez.

SZÖVEG, TÁBLÁZAT, GRAFIKON, GRAFIKA-KÉP készítéséhez sokoldalúan felhasználható a PRT—80 GS GRAFIKUS MOZAIKNYOMTATÓ

Személyi számítógépekhez, mikro-, minigépekhez intelligens terminálokhoz, mérőkészülékekhez.

A berendezést ajánljuk: PRIMÓ, IBM és IBM-kompatíbilis, valamint Sinclair és Commodore személyi számítógépekhez, amelyeknél a csatlakoztatást a megrendelő kívánságára elvégezzük.

Műszaki Vevőszolgálat:
ELEKTROMODUL
Budapest XIII.,
Victor Hugó u. 11—15.
Telefon: 495-340,
251 sz. mellék.

Árusítás:
ELEKTROMODUL
2. Sz. Szakboltjában
Budapest XIII.,
Jászai Mari tér 5.
Telefon: 530-800

Gyártja:
BHG Híradástechnikai Vállalat
1119 Budapest,
Fehérvári u. 70.
H—1509 Budapest, Pf. 2.
Telefon: 453-300

BHG®
BUDAPEST

ATARI
800 XL
Megvette már?
ZSABI and GABI
DOS 3
ATARI DISK DRIVE 1050
ATARI 1029